CODING, CRYPTOGRAPHY and CRYPTOGRAPHIC PROTOCOLS

prof. RNDr. Jozef Gruska, DrSc.

Faculty of Informatics
Masaryk University

December 6, 2011

Technické řešení této výukové pomůcky je spolufinancováno Evropským sociálním fondem a státním rozpočtem České republiky.

EVROPSKÁ UNIE

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

OP Vzdělávání pro konkurenceschopnost

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Contents

LITERATURE

- R. Hill: A first course in coding theory, Claredon Press, 1985
- V. Pless: Introduction to the theory of error-correcting codes, John Willey, 1998
- J. Gruska: Foundations of computing, Thomson International Computer Press, 1997
- A. Salomaa: Public-key cryptography, Springer, 1990
- D. R. Stinson: Cryptography: theory and practice, CRC Press, 1995
- W. Trappe, L. Washington: Introduction to cryptography with coding theory
- B. Schneier: Applied cryptography, John Willey and Sons, 1996
- J. Gruska: Quantum computing, McGraw-Hill, 1999 (For additions and updatings: `http://www.mcgraw-hill.co.uk/gruska`)
- S. Singh, The code book, Anchor Books, 1999
- D. Kahn: The codebreakers. Two story of secret writing. Macmillan, 1996 (An entertaining and informative history of cryptography.)

INTRODUCTION

- Transmission of classical information in time and space is nowadays very easy (through noiseless channel).

It took centuries, and many ingenious developments and discoveries (writing, book printing, photography, movies, telegraph, telephone, radio transmissions,TV, -sounds recording – records, tapes, discs) and the idea of the digitalisation of all forms of information to discover fully this property of information.

Coding theory develops methods to protect information against a noise.

- Information is becoming an increasingly valuable commodity for both individuals and society.

Cryptography develops methods how to ensure secrecy of information and identity, privacy or anonymity of users.

- A very important property of information is that it is often very easy to make unlimited number of copies of information.

Steganography develops methods to hide important information in innocently looking information (and that can be used to protect intellectual properties).

The history of cryptography is the story of centuries-old battles between codemakers (ciphermakers) and codebreakers (cipherbreakers), an intellectual arms race that has had a dramatic impact on the course of history.

The ongoing battle between codemakers and codebreakers has inspired a whole series of remarkable scientific breakthroughts.

History is full of ciphers. They have decided the outcomes of battles and led to the deaths of kings and queens.

Security of communication and data and identity or privacy of users are of key importance for information society. Cryptography, broadly understood, is an important tool to achieve such a goal.

Part I

Basics of coding theory

ABSTRACT

Coding theory - theory of error correcting codes - is one of the most interesting and applied part of mathematics and informatics.

All real communication systems that work with digitally represented data, as CD players, TV, fax machines, internet, satellites, mobiles, require to use error correcting codes because all real channels are, to some extent, noisy – due to interference caused by environment

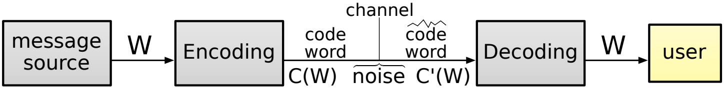
- Coding theory problems are therefore among the very basic and most frequent problems of storage and transmission of information.
- Coding theory results allow to create reliable systems out of unreliable systems to store and/or to transmit information.
- Coding theory methods are often elegant applications of very basic concepts and methods of (abstract) algebra.

This first chapter presents and illustrates the very basic problems, concepts, methods and results of coding theory.

CODING - BASIC CONCEPTS

Without coding theory and error-correcting codes there would be no deep-space travel and pictures, no satellite TV, no compact disc, no . . . no . . . no . . . .

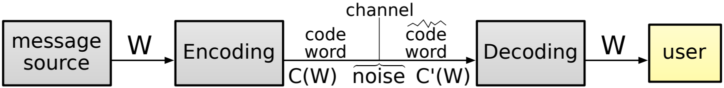
Error-correcting codes are used to correct messages when they are transmitted through noisy channels.



CODING - BASIC CONCEPTS

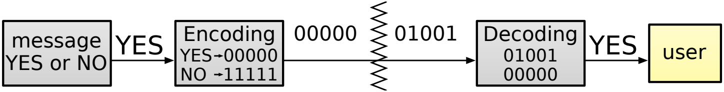
Without coding theory and error-correcting codes there would be no deep-space travel and pictures, no satellite TV, no compact disc, no ... no ... no ....

Error-correcting codes are used to correct messages when they are transmitted through noisy channels.



Error correcting framework

Example



A code C over an alphabet Σ is a subset of $\Sigma^*(C \subseteq \Sigma^*)$.

A q-nary code is a code over an alphabet of q-symbols.

A binary code is a code over the alphabet $\{0, 1\}$.

Examples of codes $C1 = \{00, 01, 10, 11\}$ $C2 = \{000, 010, 101, 100\}$

$C3 = \{00000, 01101, 10111, 11011\}$

CHANNEL

is any physical medium through which information is transmitted.
(Telephone lines and the atmosphere are examples of channels.)

CHANNEL

is any physical medium through which information is transmitted.
(Telephone lines and the atmosphere are examples of channels.)

NOISE

may be caused by sunspots, lighting, meteor showers, random radio disturbance, poor typing, poor hearing, ...

CHANNEL

is any physical medium through which information is transmitted.
(Telephone lines and the atmosphere are examples of channels.)

NOISE

may be caused by sunspots, lighting, meteor showers, random radio disturbance, poor typing, poor hearing, ...

TRANSMISSION GOALS

1. Fast encoding of information.
2. Easy transmission of encoded messages.
3. Fast decoding of received messages.
4. Reliable correction of errors introduced in the channel.
5. Maximum transfer of information per unit time.

CHANNEL

is any physical medium through which information is transmitted.
(Telephone lines and the atmosphere are examples of channels.)

NOISE

may be caused by sunspots, lighting, meteor showers, random radio disturbance, poor typing, poor hearing, ...

TRANSMISSION GOALS

1. Fast encoding of information.
2. Easy transmission of encoded messages.
3. Fast decoding of received messages.
4. Reliable correction of errors introduced in the channel.
5. Maximum transfer of information per unit time.

BASIC METHOD OF FIGHTING ERRORS: REDUNDANCY!!!

0 is encoded as 00000 and 1 is encoded as 11111.

IMPORTANCE of ERROR-CORRECTING CODES

In a good cryptosystem a change of a single bit of the cryptotext should change so many bits of the plaintext obtained from the cryptotext that the plaintext gets uncomprehensible.

Methods to detect and correct errors when cryptotexts are transmitted are therefore much needed.

Also many non-cryptographic applications require error-correcting codes. For example, mobiles, CD-players,...

BASIC IDEA

The details of techniques used to protect information against noise in practice are sometimes rather complicated, but basic principles are easily understood.

The key idea is that in order to protect a message against a noise, we should encode the message by adding some redundant information to the message.

In such a case, even if the message is corrupted by a noise, there will be enough redundancy in the encoded message to recover – to decode the message completely.

EXAMPLE

In case of the encoding

$$0 \to 000 \quad 1 \to 111$$

the probability of the bit error $p \leq \frac{1}{2}$, and the majority voting decoding

$$000, 001, 010, 100 \to 000 \quad \text{and} \quad 111, 110, 101, 011 \to 111$$

the probability of an erroneous decoding (if there are 2 or 3 errors) is

$$3p^2(1-p) + p^3 = 3p^2 - 2p^3 < p$$

Story Alice and Bob share an identical map (Fig. 1) gridded as shown in Fig.1. Only Alice knows the route through which Bob can reach her avoiding the enemy territory. Alice wants to send Bob the following information about the safe route he should take.

NNWNNWWSSWWNNNNWWN

Three ways to encode the safe route from Bob to Alice are:

1. $C1 = \{N = 00, W = 01, S = 11, E = 10\}$

Any error in the code word

000001000001011111010100000000010100

would be a disaster.

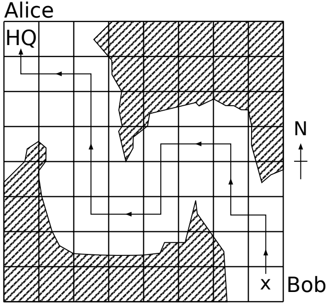


Fig. 1

EXAMPLE: Coding of a path avoiding an enemy territory

Story Alice and Bob share an identical map (Fig. 1) gridded as shown in Fig.1. Only Alice knows the route through which Bob can reach her avoiding the enemy territory. Alice wants to send Bob the following information about the safe route he should take.

NNWNNWWSSWWNNNNWWN

Three ways to encode the safe route from Bob to Alice are:

1. $C1 = \{N = 00, W = 01, S = 11, E = 10\}$

Any error in the code word

000001000001011111010100000000010100

would be a disaster.

2. $C2 = \{000, 011, 101, 110\}$

A single error in encoding each of symbols N, W, S, E can be detected.

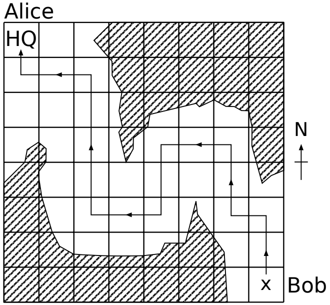


Fig. 1

EXAMPLE: Coding of a path avoiding an enemy territory

Story Alice and Bob share an identical map (Fig. 1) gridded as shown in Fig.1. Only Alice knows the route through which Bob can reach her avoiding the enemy territory. Alice wants to send Bob the following information about the safe route he should take.

NNWNNWWSSWWNNNNWWN

Three ways to encode the safe route from Bob to Alice are:

1. $C1 = \{N = 00, W = 01, S = 11, E = 10\}$

Any error in the code word

000001000001011111010100000000010100

would be a disaster.

2. $C2 = \{000, 011, 101, 110\}$

A single error in encoding each of symbols N, W, S, E can be detected.

3. $C3 = \{00000, 01101, 10110, 11011\}$

A single error in decoding each of symbols N, W, S, E can be corrected.

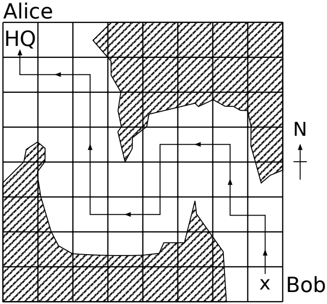


Fig. 1

Block code - a code with all words of the same length.
Codewords - words of some code.

Basic terminology

Block code - a code with all words of the same length.
Codewords - words of some code.

Basic assumptions about channels

1. Code length preservation Each output word of a channel has the same length as the input codeword.
2. Independence of errors The probability of any one symbol being affected in transmissions is the same.

Basic terminology

Block code - a code with all words of the same length.
Codewords - words of some code.

Basic assumptions about channels

1. Code length preservation Each output word of a channel has the same length as the input codeword.
2. Independence of errors The probability of any one symbol being affected in transmissions is the same.

Basic strategy for decoding

For decoding we use the so-called maximal likehood principle, or nearest neighbor decoding strategy, or majority voting decoding strategy which says that the receiver should decode a word w' as that codeword w that is the closest one to w'.

HAMMING DISTANCE

The intuitive concept of "**closeness**" of two words is well formalized through Hamming distance $h(x, y)$ of words x, y. For two words x, y

h(x, y) = the number of symbols in which the words x and y differ.

Example: $h(10101, 01100) = 3$, $h(fourth, eighth) = 4$

HAMMING DISTANCE

The intuitive concept of "**closeness**" of two words is well formalized through Hamming distance $h(x, y)$ of words x, y. For two words x, y

h(x, y) = the number of symbols in which the words x and y differ.

Example: $h(10101, 01100) = 3$, $h(fourth, eighth) = 4$

Properties of Hamming distance

1. $h(x, y) = 0 \Leftrightarrow x = y$
2. $h(x, y) = h(y, x)$
3. $h(x, z) \leq h(x, y) + h(y, z)$ triangle inequality

An important parameter of codes C is their minimal distance.

$$h(C) = min\{h(x, y) \,|\, x, y \in C, x \neq y\},$$

because $h(C)$ is the smallest number of errors needed to change one codeword into another.

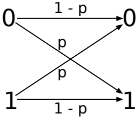
Theorem Basic error correcting theorem

1. A code C can detect up to s errors if $h(C) \geq s + 1$.
2. A code C can correct up to t errors if $h(C) \geq 2t + 1$.

Proof (1) Trivial. (2) Suppose $h(C) \geq 2t + 1$. Let a codeword x is transmitted and a word y is recceived with $h(x, y) \leq t$. If $x' \neq x$ is a codeword, then $h(y, x') \geq t + 1$ because otherwise $h(y, x') < t + 1$ and therefore $h(x, x') \leq h(x, y) + h(y, x') < 2t + 1$ what contradicts the assumption $h(C) \geq 2t + 1$.

BINARY SYMMETRIC CHANNEL

Consider a transition of binary symbols such that each symbol has probability of error $p < \frac{1}{2}$.



Binary symmetric channel

If n symbols are transmitted, then the probability of t errors is

$$p^t(1-p)^{n-t}\binom{n}{t}$$

In the case of binary symmetric channels, the "nearest neighbour decoding strategy" is also "maximum likelihood decoding strategy".

Example Consider $C = \{000, 111\}$ and the nearest neighbour decoding strategy. Probability that the received word is decoded correctly

$$\text{as } 000 \text{ is } (1-p)^3 + 3p(1-p)^2,$$
$$\text{as } 111 \text{ is } (1-p)^3 + 3p(1-p)^2,$$

Therefore $$P_{err}(C) = 1 - ((1-p)^3 + 3p(1-p)^2)$$
is probability of erroneous decoding.

Example If $p = 0.01$, then $P_{err}(C) = 0.000298$ and only one word in 3356 will reach the user with an error.

POWER of PARITY BITS

Example Let all 2^{11} of binary words of length 11 be codewords.
Let the probability p of a bit error be 10^{-8}.
Let bits be transmitted at the rate 10^7 bits per second.
The probability that a word is transmitted incorrectly is approximately

$$11p(1-p)^{10} \approx \frac{11}{10^8}.$$

Therefore $\frac{11}{10^8} \cdot \frac{10^7}{11} = 0.1$ of words per second are transmitted incorrectly.
One wrong word is transmitted every 10 seconds, 360 erroneous words every hour and 8640 words every day without being detected!
Let now one parity bit be added.
Any single error can be detected!!!
The probability of at least two errors is:

$$1-(1-p)^{12}-12(1-p)^{11}p \approx \binom{12}{2}(1-p)^{10}p^2 \approx \frac{66}{10^{16}}$$

Therefore approximately $\frac{66}{10^{16}} \cdot \frac{10^7}{12} \approx 5.5 \cdot 10^{-9}$ words per second are transmitted with an undetectable error.
Corollary One undetected error occurs only every 2000 days! $(2000 \approx \frac{10^9}{5.5 \times 86400})$.

TWO-DIMENSIONAL PARITY CODE

The two-dimensional parity code arranges the data into a two-dimensional array and then to each row (column) parity bit is attached.
Example Binary string

$$10001011000100101111$$

is represented and encoded as follows

$$\begin{array}{ccccc} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{array} \rightarrow \begin{array}{cccccc} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \end{array}$$

Question How much better is two-dimensional encoding than one-dimensional encoding?

Notation: An (n, M, d)-code C is a code such that

- n - is the length of codewords.
- M - is the number of codewords.
- d - is the minimum distance in C.

NOTATIONS and EXAMPLES

Notation: An (n, M, d)-code C is a code such that

- n - is the length of codewords.
- M - is the number of codewords.
- d - is the minimum distance in C.

Example:
$C1 = \{00, 01, 10, 11\}$ is a (2,4,1)-code.
$C2 = \{000, 011, 101, 110\}$ is a (3,4,2)-code.
$C3 = \{00000, 01101, 10110, 11011\}$ is a (5,4,3)-code.

Comment: A good (n, M, d)-code has small n and large M and d.

EXAMPLES from DEEP SPACE TRAVELS

Examples (Transmission of photographs from the deep space)

- In 1965-69 **Mariner 4-5** took the first photographs of another planet - 22 photos. Each photo was divided into 200×200 elementary squares - pixels. Each pixel was assigned 6 bits representing 64 levels of brightness. Hadamard code was used.

  Transmission rate: 8.3 bits per second.

- In 1970-72 **Mariners 6-8** took such photographs that each picture was broken into 700×832 squares. Reed-Muller (32,64,16) code was used.

  Transmission rate was 16200 bits per second. (Much better pictures)

HADAMARD CODE

In Mariner 5, 6-bit pixels were encoded using 32-bit long Hadamard code that could correct up to 7 errors.

Hadamard code has 64 codewords. 32 of them are represented by the 32×32 matrix $H = \{h_{IJ}\}$, where $0 \leq i, j \leq 31$ and

$$h_{ij} = (-1)^{a_0 b_0 + a_1 b_1 + \ldots + a_4 b_4}$$

where i and j have binary representations

$$i = a_4 a_3 a_2 a_1 a_0,\ j = b_4 b_3 b_2 b_1 b_0$$

The remaing 32 codewords are represented by the matrix $-H$.
Decoding is quite simple.

CODE RATE

For q-nary (n, M, d)-code we define code rate, or information rate, R, by

$$R = \frac{lg_q M}{n}.$$

The code rate represents the ratio of the number of needed input data symbols to the number of transmitted code symbols.

Code rate (6/32 for Hadamard code), is an important parameter for real implementations, because it shows what fraction of the bandwidth is being used to transmit actual data.

The ISBN-code I

Each book till 1.1.2007 had International Standard Book Number which was a 10-digit codeword produced by the publisher with the following structure:

l	p	m	w	$= x_{10} \dots x_1$
language	publisher	number	weighted check sum	
0	07	709503	0	

such that $\sum_{i=1}^{10} ix_i \equiv 0 \ (mod 11)$

The publisher has to put $x_1 = X$ if x_1 is to be 10.

The ISBN code was designed to detect: (a) any single error (b) any double error created by a transposition

The ISBN-code I

Each book till 1.1.2007 had International Standard Book Number which was a 10-digit codeword produced by the publisher with the following structure:

l	p	m	w	$= x_{10} \dots x_1$
language	publisher	number	weighted check sum	
0	07	709503	0	

such that $\sum_{i=1}^{10} ix_i \equiv 0 \ (mod 11)$

The publisher has to put $x_1 = X$ if x_1 is to be 10.
The ISBN code was designed to detect: (a) any single error (b) any double error created by a transposition

Single error detection

Let $X = x_{10} \dots x_1$ be a correct code and let

$$Y = x_{10} \dots x_{j+1} y_j x_{j-1} \dots x_1 \text{ with } y_J = x_J + a, a \neq 0$$

In such a case:

$$\sum_{i=1}^{10} iy_i = \sum_{i=1}^{10} ix_i + ja \neq 0 \ (mod 11)$$

Transposition detection

Let x_J and x_k be exchanged.

$$\sum_{i=1}^{10} iy_i = \sum_{i=1}^{10} ix_i + (k-j)x_j + (j-k)x_k = (k-j)(x_j - x_k) \neq 0 \ (mod 11)$$

if $k \neq j$ and $x_j \neq x_k$.

New ISBN code

Starting 1.1.2007 instead of 10-digit ISBN code a 13-digit ISBN code is being used.

New ISBN number can be obtained from the old one by preceeding the old code with three digits 978.

For details about 13-digit ISBN see

`http://www.en.wikipedia.org/Wiki/International_Standard_Book_Number`

EQUIVALENCE of CODES

Definition Two q-ary codes are called equivalent if one can be obtained from the other by a combination of operations of the following type:

(a) a permutation of the positions of the code.

(b) a permutation of symbols appearing in a fixed position.

Question: Let a code be displayed as an $M \times n$ matrix. To what correspond operations (a) and (b)?

Claim: Distances between codewords are unchanged by operations (a), (b).

Consequently, equivalent codes have the same parameters (n,M,d) (and correct the same number of errors).

EQUIVALENCE of CODES

Definition Two q-ary codes are called equivalent if one can be obtained from the other by a combination of operations of the following type:

(a) a permutation of the positions of the code.

(b) a permutation of symbols appearing in a fixed position.

Question: Let a code be displayed as an M $\times$ n matrix. To what correspond operations (a) and (b)?

Claim: Distances between codewords are unchanged by operations (a), (b). Consequently, equivalent codes have the same parameters (n,M,d) (and correct the same number of errors).

Examples of equivalent codes

$$(1)\ \begin{Bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \end{Bmatrix} \begin{Bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \end{Bmatrix} \ (2)\ \begin{Bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 2 & 2 & 2 \end{Bmatrix} \begin{Bmatrix} 0 & 1 & 2 \\ 1 & 2 & 0 \\ 2 & 0 & 1 \end{Bmatrix}$$

Lemma Any q-ary (n, M, d)-code over an alphabet $\{0, 1, \ldots, q-1\}$ is equivalent to an (n, M, d)-code which contains the all-zero codeword $00\ldots0$.

Proof Trivial.

THE MAIN CODING THEORY PROBLEM

A good (n, M, d)-code has small n, large M and large d.

The main coding theory problem is to optimize one of the parameters n, M, d for given values of the other two.

Notation: $A_q(n, d)$ is the largest M such that there is an q-nary (n, M, d)-code.

Theorem

(a) $A_q(n, 1) = q^n$;

(b) $A_q(n, n) = q$.

Proof

(a) obvious;

(b) Let C be an q-nary (n, M, n)-code. Any two distinct codewords of C differ in all n positions. Hence symbols in any fixed position of M codewords have to be different $\Rightarrow A_q(n, n) \leq q$. Since the q-nary repetition code is (n, q, n)-code, we get $A_q(n, n) \geq q$.

EXAMPLE

Example Proof that $A_2(5,3) = 4$.

(a) Code C_3 is a $(5,4,3)$-code, hence $A_2(5,3) \geq 4$.

(b) Let C be a $(5,M,3)$-code with $M = 5$.

- By previous lemma we can assume that $00000 \in C$.
- C has to contain at most one codeword with at least four 1's. (otherwise $d(x,y) \leq 2$ for two such codewords x, y)
- Since $00000 \in C$, there can be no codeword in C with at most one or two 1.
- Since $d = 3$, C cannot contain three codewords with three 1's.
- Since $M \geq 4$, there have to be in C two codewords with three 1's. (say 11100, 00111), the only possible codeword with four or five 1's is then 11011.

DESIGN of ONE CODE from ANOTHER ONE

Theorem Suppose d is odd. Then a binary (n, M, d)-code exists if a binary $(n+1, M, d+1)$-code exists.

Proof Only if case: Let C be a binary (n, M, d) code. Let

$$C' = \left\{x_1 \dots x_n x_{n+1} \middle| x_1 \dots x_n \in C, x_{n+1} = \left(\sum_{i=1}^{n} x_i\right) \mathit{mod}\ 2\right\}$$

Since parity of all codewords in C' is even, $d(x', y')$ is even for all

$$x', y' \in C'.$$

Hence $d(C')$ is even. Since $d \leq d(C') \leq d+1$ and d is odd,

$$d(C') = d+1.$$

Hence C' is an $(n+1, M, d+1)$-code.

If case: Let D be an $(n+1, M, d+1)$-code. Choose code words x, y of D such that $d(x, y) = d+1$.
Find a position in which x, y differ and delete this position from all codewords of D.
Resulting code is an (n, M, d)-code.

A COROLLARY

Corollary:
If d is odd, then $A_2(n,d) = A_2(n+1, d+1)$.
If d is even, then $A_2(n,d) = A_2(n-1, d-1)$.

Example $\quad A_2(5,3) = 4 \Rightarrow A_2(6,4) = 4$
$(5,4,3)$-code $\Rightarrow$ $(6,4,4)$-code

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \end{matrix} \quad \text{by adding check.}$$

A SPEHERE and its CONTENTS

Notation F_q^n - is a set of all words of length n over the alphabet $\{0, 1, 2, \ldots, q-1\}$

Definition For any codeword $u \in F_q^n$ and any integer $r \geq 0$ the sphere of radius r and centre u is denoted by

$$S(u, r) = \{v \in F_q^n | h(u, v) \leq r\}.$$

Theorem A sphere of radius r in F_q^n, $0 \leq r \leq n$ contains

$$\binom{n}{0} + \binom{n}{1}(q-1) + \binom{n}{2}(q-1)^2 + \ldots + \binom{n}{r}(q-1)^r$$

words.

Notation F_q^n - is a set of all words of length n over the alphabet $\{0, 1, 2, \ldots, q-1\}$

Definition For any codeword $u \in F_q^n$ and any integer $r \geq 0$ the sphere of radius r and centre u is denoted by

$$S(u, r) = \{v \in F_q^n | h(u, v) \leq r\}.$$

Theorem A sphere of radius r in F_q^n, $0 \leq r \leq n$ contains

$$\binom{n}{0} + \binom{n}{1}(q-1) + \binom{n}{2}(q-1)^2 + \ldots + \binom{n}{r}(q-1)^r$$

words.

Proof Let u be a fixed word in F_q^n. The number of words that differ from u in m positions is

$$\binom{n}{m}(q-1)^m.$$

GENERAL UPPER BOUNDS

Theorem **(The sphere-packing or Hamming bound)**
If C is a q-nary $(n, M, 2t+1)$-code, then

$$M\left\{\binom{n}{0}+\binom{n}{1}(q-1)+\ldots+\binom{n}{t}(q-1)^{t}\right\} \leq q^{n} \tag{1}$$

Proof Any two spheres of radius t centred on distinct codewords have no codeword in common. Hence the total number of words in M spheres of radius t centred on M codewords is given by the left side (1). This number has to be less or equal to q^n.

A code which achieves the sphere-packing bound from (1), i.e. such a code that equality holds in (1), is called a **perfect code**.

Singleton bound: If C is an q-ary (n, M, d) code, then

$$M \leq q^{n-d+1}$$

Example An $(7, M, 3)$-code is perfect if

$$M\left(\binom{7}{0}+\binom{7}{1}\right)=2^7$$

i.e. $M = 16$

An example of such a code:

$C4 = \{0000000, 1111111, 1000101, 1100010, 0110001, 1011000, 0101100,$
$0010110, 0001011, 0111010, 0011101, 1001110, 0100111, 1010011, 1101001, 1110100\}$

Table of $A_2(n,d)$ from 1981

n	$d=3$	$d=5$	$d=7$
5	4	2	-
6	8	2	-
7	16	2	2
8	20	4	2
9	40	6	2
10	72-79	12	2
11	144-158	24	4
12	256	32	4
13	512	64	8
14	1024	128	16
15	2048	256	32
16	2560-3276	256-340	36-37

For current best results see http://www.codetables.de

LOWER BOUND for $A_q(n,d)$

The following lower bound for $A_q(n,d)$ is known as Gilbert-Varshamov bound:

Theorem Given $d \leq n$, there exists a q-ary (n,M,d)-code with

$$M \geq \frac{q^n}{\sum_{j=0}^{d-1} \binom{n}{j}(q-1)^j}$$

and therefore

$$A_q(n,d) \geq \frac{q^n}{\sum_{j=0}^{d-1} \binom{n}{j}(q-1)^j}$$

ERROR DETECTION

Error detection is much more modest aim than error correction.

Error detection is suitable in the cases that channel is so good that probability of error is small and if an error is detected, the receiver can ask to renew the transmission.

For example, two main requirements for many telegraphy codes used to be:

- Any two codewords had to have distance at least 2;
- No codeword could be obtained from another codeword

by transposition of two adjacent letters.

Pictures of Saturn taken by Voyager

Pictures of Saturn taken by Voyager, in 1980, had 800×800 pixels with 8 levels of brightness.

Since pictures were in color, each picture was transmitted three times; each time through different color filter. The full color picture was represented by

$$3 \times 800 \times 800 \times 8 = 13360000 \text{ bits.}$$

To transmit pictures Voyager used the Golay code G_{24}.

GENERAL CODING PROBLEM

Important problems of information theory are how to define formally such concepts as information and how to store or transmit information efficiently.

Let X be a random variable (source) which takes any value x with probability $p(x)$. The entropy of X is defined by

$$S(X) = -\sum_x p(x) lg\ p(x)$$

and it is considered to be the information content of X.

In a special case of a binary variable X which takes on the value 1 with probability p and the value 0 with probability $1 - p$

$$S(X) = H(p) = -p\ lg\ p - (1-p) lg(1-p)$$

Problem: What is the minimal number of bits needed to transmit n values of X?
Basic idea: To encode more probable outputs of X by shorter binary words.
Example (Morse code - 1838)

a .-	b -...	c -.-.	d -..	e .	f ..-.	g –.
h	i ..	j .—	k -.-	l .-..	m –	n -.
o —	p .–.	q –.-	r .-.	s ...	t -	u ..-
v ...-	w .–	x -..-	y -.–	z –..		

Shannon's noiseless coding theorem says that in order to transmit n values of X, we need, and it is sufficient, to use $nS(X)$ bits.

More exactly, we cannot do better than the bound $nS(X)$ says, and we can reach the bound $nS(X)$ as close as desirable.

Example Let a source X produce the value 1 with probability $p = \frac{1}{4}$
and the value 0 with probability $1 - p = \frac{3}{4}$

Assume we want to encode blocks of the outputs of X of length 4.

By Shannon's theorem we need $4H(\frac{1}{4}) = 3.245$ bits per blocks (in average)

A simple and practical method known as **Huffman code** requires in this case 3.273 bits per a 4-bit message.

mess.	code	mess.	code	mess.	code	mess.	code
0000	10	0100	010	1000	011	1100	11101
0001	000	0101	11001	1001	11011	1101	111110
0010	001	0110	11010	1010	11100	1110	111101
0011	11000	0111	1111000	1011	111111	1111	1111001

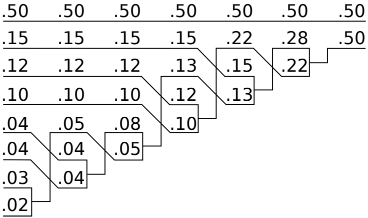
Observe that this is a **prefix code** - no codeword is a prefix of another codeword.

DESIGN of HUFFMAN CODE II

Given a sequence of n objects, $x_1, \ldots, x_n$ with probabilities $p_1 \geq \ldots \geq p_n$.

Stage 1 - shrinking of the sequence.

- Replace x_{n-1}, x_n with a new object y_{n-1} with probability $p_{n-1} + p_n$ and rearrange sequence so one has again non-increasing probabilities.
- Keep doing the above step till the sequence shrinks to two objects.



DESIGN of HUFFMAN CODE II

Given a sequence of n objects, $x_1, \dots, x_n$ with probabilities $p_1 \geq \dots \geq p_n$.

Stage 1 - shrinking of the sequence.

- Replace x_{n-1}, x_n with a new object y_{n-1} with probability $p_{n-1} + p_n$ and rearrange sequence so one has again non-increasing probabilities.
- Keep doing the above step till the sequence shrinks to two objects.

.50	.50	.50	.50	.50	.50	.50
.15	.15	.15	.15	.22	.28	.50
.12	.12	.12	.13	.15	.22	
.10	.10	.10	.12	.13		
.04	.05	.08	.10			
.04	.04	.05				
.03	.04					
.02						

Stage 2 - extending the code - Apply again and again the following method.

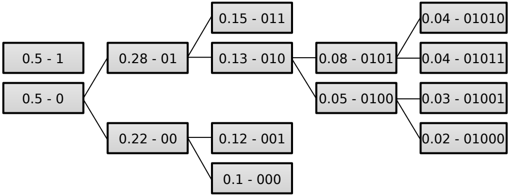
If $C = \{c_1, \dots, c_r\}$ is a prefix optimal code for a source S_r, then $C' = \{c'_1, \dots, c'_{r+1}\}$ is an optimal code for S_{r+1}, where

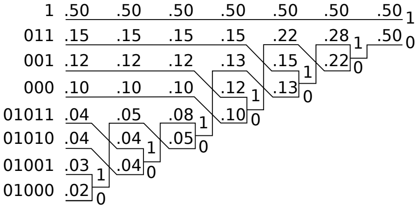
$$c'_i = c_i \quad 1 \leq i \leq r-1$$
$$c'_r = c_r 1$$
$$c'_{r+1} = c_r 0.$$

Stage 2 Apply again and again the following method:

If $C = \{c_1, \ldots, c_r\}$ is a prefix optimal code for a source S_r, then $C' = \{c'_1, \ldots, c'_{r+1}\}$ is an optimal code for S_{r+1}, where

$$c'_i = c_i \quad 1 \leq i \leq r-1$$
$$c'_r = c_r 1$$
$$c'_{r+1} = c_r 0.$$





A BIT OF HISTORY I

The subject of error-correcting codes arose originally as a response to practical problems in the reliable communication of digitally encoded information.

The discipline was initiated in the paper

Claude Shannon: A mathematical theory of communication, Bell Syst.Tech. Journal V27, 1948, 379-423, 623-656

Shannon's paper started the scientific discipline **information theory** and **error-correcting codes** are its part.

Originally, information theory was a part of electrical engineering. Nowadays, it is an important part of mathematics and also of informatics.

SHANNON's VIEW

In the introduction to his seminal paper "A mathematical theory of communication" Shannon wrote:

The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point.

Part II

Linear codes

ABSTRACT

Most of the important codes are special types of so-called linear codes.

Linear codes are of very large importance because they have
very concise description,
very nice properties,
very easy encoding
and,
in principle, easy to describe decoding.

LINEAR CODES

Linear codes are special sets of words of the length n over an alphabet $\Sigma_q = \{0, .., q-1\}$, where q is a power of prime. Since now on F_q^n will be the vector spaces of all n-tuples over the finite field F_q (on the set $\{0, .., q-1\}$ and arithmetical operations modulo q.)

Definition A subset $C \subseteq V(n, q)$ is a linear code if

1. $u + v \in C$ for all $u, v \in C$
2. $au \in C$ for all $u \in C, a \in GF(q)$ - $\{$Galoi field over $\Sigma_q\}$

Example Codes C_1, C_2, C_3 introduced in Lecture 1 are linear codes.

LINEAR CODES

Linear codes are special sets of words of the length n over an alphabet $\Sigma_q = \{0, .., q-1\}$, where q is a power of prime. Since now on F_q^n will be the vector spaces of all n-tuples over the finite field F_q (on the set $\{0, .., q-1\}$ and arithmetical operations modulo q.)

Definition A subset $C \subseteq V(n, q)$ is a linear code if

1. $u + v \in C$ for all $u, v \in C$
2. $au \in C$ for all $u \in C, a \in GF(q)$ - $\{$Galoi field over $\Sigma_q\}$

Example Codes C_1, C_2, C_3 introduced in Lecture 1 are linear codes.

Lemma A subset $C \subseteq V(n, q)$ is a linear code iff one of the following conditions is satisfied

1. C is a subspace of $V(n, q)$
2. sum of any two codewords from C is in C (for the case $q = 2$)

If C is a k-dimensional subspace of $V(n, q)$, then C is called $[n, k]$-code. It has q^k codewords if q is prime. If minimal distance of C is d, then it is called $[n, k, d]$ code.

Linear codes are also called "group codes".

EXERCISE

Which of the following binary codes are linear?

$C_1 = \{00, 01, 10, 11\}$
$C_2 = \{000, 011, 101, 110\}$
$C_3 = \{00000, 01101, 10110, 11011\}$
$C_5 = \{101, 111, 011\}$
$C_6 = \{000, 001, 010, 011\}$
$C_7 = \{0000, 1001, 0110, 1110\}$

EXERCISE

Which of the following binary codes are linear?

$C_1 = \{00, 01, 10, 11\}$
$C_2 = \{000, 011, 101, 110\}$
$C_3 = \{00000, 01101, 10110, 11011\}$
$C_5 = \{101, 111, 011\}$
$C_6 = \{000, 001, 010, 011\}$
$C_7 = \{0000, 1001, 0110, 1110\}$

How to create a linear code

Notation If S is a set of vectors of a vector space, then let $\langle S \rangle$ be the set of all linear combinations of vectors from S.

Theorem For any subset S of a linear space, $\langle S \rangle$ is a linear space that consists of the following words:

- the zero word,
- all words in S,
- all sums of two or more words in S.

EXERCISE

Which of the following binary codes are linear?

$C_1 = \{00, 01, 10, 11\}$
$C_2 = \{000, 011, 101, 110\}$
$C_3 = \{00000, 01101, 10110, 11011\}$
$C_5 = \{101, 111, 011\}$
$C_6 = \{000, 001, 010, 011\}$
$C_7 = \{0000, 1001, 0110, 1110\}$

How to create a linear code

Notation If S is a set of vectors of a vector space, then let $\langle S \rangle$ be the set of all linear combinations of vectors from S.

Theorem For any subset S of a linear space, $\langle S \rangle$ is a linear space that consists of the following words:

- the zero word,
- all words in S,
- all sums of two or more words in S.

Example

$$S = \{0100, 0011, 1100\}$$
$$\langle S \rangle = \{0000, 0100, 0011, 1100, 0111, 1011, 1000, 1111\}.$$

BASIC PROPERTIES of LINEAR CODES I

Notation: w(x) (weight of x) denotes the number of non-zero entries of x.

Lemma If $x, y \in V(n, q)$, then $h(x, y) = w(x - y)$.

Proof $x - y$ has non-zero entries in exactly those positions where x and y differ.

BASIC PROPERTIES of LINEAR CODES I

Notation: w(x) (weight of x) denotes the number of non-zero entries of x.

Lemma If $x, y \in V(n, q)$, then $h(x, y) = w(x - y)$.

Proof $x - y$ has non-zero entries in exactly those positions where x and y differ.

Theorem Let C be a linear code and let weight of C, notation $w(C)$, be the smallest of the weights of non-zero codewords of C. Then $h(C) = w(C)$.

Proof There are $x, y \in C$ such that $h(C) = h(x, y)$. Hence $h(C) = w(x - y) \geq w(C)$.

On the other hand, for some $x \in C$

$$w(C) = w(x) = h(x, 0) \geq h(C).$$

Consequence

- If C is a code with m codewords, then in order to determine $h(C)$ one has to make $\binom{m}{2} = \Theta(m^2)$ comparisons in the worst case.
- If C is a linear code, then in order to compute $h(C)$, $m - 1$ comparisons are enough.

If C is a linear $[n, k]$-code, then it has a basis consisting of k codewords.

Example

Code

$$C_4 = \{0000000, 1111111, 1000101, 1100010, \\ 0110001, 1011000, 0101100, 0010110, \\ 0001011, 0111010, 0011101, 1001110, \\ 0100111, 1010011, 1101001, 1110100\}$$

has the basis

$$\{1111111, 1000101, 1100010, 0110001\}.$$

How many different bases has a linear code?

BASIC PROPERTIES of LINEAR CODES II

If C is a linear $[n, k]$-code, then it has a basis consisting of k codewords.

Example

Code

$$\begin{aligned} C_4 \quad = \quad & \{0000000, 1111111, 1000101, 1100010, \\ & 0110001, 1011000, 0101100, 0010110, \\ & 0001011, 0111010, 0011101, 1001110, \\ & 0100111, 1010011, 1101001, 1110100\} \end{aligned}$$

has the basis

$$\{1111111, 1000101, 1100010, 0110001\}.$$

How many different bases has a linear code?

Theorem A binary linear code of dimension k has

$$\frac{1}{k!} \prod_{i=0}^{k-1} (2^k - 2^i)$$

bases.

ADVANTAGES and DISADVANTAGES of LINEAR CODES I.

Advantages - big.

1. Minimal distance $h(C)$ is easy to compute if C is a linear code.
2. Linear codes have simple specifications.

- To specify a non-linear code usually all codewords have to be listed.
- To specify a linear $[n, k]$-code it is enough to list k codewords (of a basis).

Definition A $k \times n$ matrix whose rows form a basis of a linear $[n, k]$-code (subspace) C is said to be the generator matrix of C.

Example The generator matrix of the code

$$C_2 = \begin{Bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{Bmatrix} \text{ is } \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$$

and of the code

$$C_4 = \text{ is } \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

3. There are simple encoding/decoding procedures for linear codes.

Disadvantages of linear codes are small:

1. Linear q-codes are not defined unless q is a prime power.
2. The restriction to linear codes might be a restriction to weaker codes than sometimes desired.

EQUIVALENCE of LINEAR CODES I

Definition Two linear codes on $GF(q)$ are called equivalent if one can be obtained from another by the following operations:

(a) permutation of the words or positions of the code;

(b) multiplication of symbols appearing in a fixed position by a non-zero scalar.

EQUIVALENCE of LINEAR CODES I

Definition Two linear codes on $GF(q)$ are called equivalent if one can be obtained from another by the following operations:

(a) permutation of the words or positions of the code;

(b) multiplication of symbols appearing in a fixed position by a non-zero scalar.

Theorem Two $k \times n$ matrices generate equivalent linear $[n, k]$-codes over $GF(q)$ if one matrix can be obtained from the other by a sequence of the following operations:

(a) permutation of the rows

(b) multiplication of a row by a non-zero scalar

(c) addition of one row to another

(d) permutation of columns

(e) multiplication of a column by a non-zero scalar

EQUIVALENCE of LINEAR CODES I

Definition Two linear codes on $GF(q)$ are called equivalent if one can be obtained from another by the following operations:

(a) permutation of the words or positions of the code;

(b) multiplication of symbols appearing in a fixed position by a non-zero scalar.

Theorem Two $k \times n$ matrices generate equivalent linear $[n, k]$-codes over $GF(q)$ if one matrix can be obtained from the other by a sequence of the following operations:

(a) permutation of the rows

(b) multiplication of a row by a non-zero scalar

(c) addition of one row to another

(d) permutation of columns

(e) multiplication of a column by a non-zero scalar

Proof Operations (a) - (c) just replace one basis by another. Last two operations convert a generator matrix to one of an equivalent code.

EQUIVALENCE of LINEAR CODES II

Theorem Let G be a generator matrix of an $[n, k]$-code. Rows of G are then linearly independent .By operations (a) - (e) the matrix G can be transformed into the form: $[I_k|A]$ where I_k is the $k \times k$ identity matrix, and A is a $k \times (n-k)$ matrix.

Theorem Let G be a generator matrix of an $[n, k]$-code. Rows of G are then linearly independent .By operations (a) - (e) the matrix G can be transformed into the form: $[I_k|A]$ where I_k is the $k \times k$ identity matrix, and A is a $k \times (n-k)$ matrix.

Example

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} \rightarrow$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} \rightarrow$$

ENCODING with LINEAR CODES

is a vector $\times$ matrix multiplication

Let C be a linear $[n, k]$-code over $GF(q)$ with a generator matrix G.

Theorem C has q^k codewords.

Proof Theorem follows from the fact that each codeword of C can be expressed uniquely as a linear combination of the basis vectors.

Corollary The code C can be used to encode uniquely q^k messages.

Let us identify messages with elements $V(k, q)$.

Encoding of a message $u = (u_1, \ldots, u_k)$ with the code C:

$u \cdot G = \sum_{i=1}^{k} u_i r_i$ where $r_1, \ldots, r_k$ are rows of G.

ENCODING with LINEAR CODES

is a vector $\times$ matrix multiplication

Let C be a linear $[n,k]$-code over $GF(q)$ with a generator matrix G.

Theorem C has q^k codewords.

Proof Theorem follows from the fact that each codeword of C can be expressed uniquely as a linear combination of the basis vectors.

Corollary The code C can be used to encode uniquely q^k messages.

Let us identify messages with elements $V(k,q)$.

Encoding of a message $u = (u_1, \ldots, u_k)$ with the code C:

$$u \cdot G = \sum_{i=1}^{k} u_i r_i \text{ where } r_1, \ldots, r_k \text{ are rows of } G.$$

Example Let C be a $[7,4]$-code with the generator matrix

$$\text{G}= \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

A message (u_1, u_2, u_3, u_4) is encoded as:???

For example:

0 0 0 0 is encoded as . . . . ?

1 0 0 0 is encoded as . . . . ?

1 1 1 0 is encoded as . . . . ?

with linear codes

Theorem If $G = \{w_i\}_{i=1}^{k}$ is a generator matrix of a binary linear code C of length n and dimension k, then

$$v = uG$$

ranges over all 2^k codewords of C as u ranges over all 2^k words of length k.
Therefore

$$C = \{uG | u \in \{0,1\}^k\}$$

Moreover

$$u_1 G = u_2 G$$

if and only if

$$u_1 = u_2.$$

Proof If $u_1 G - u_2 G = 0$, then

$$0 = \sum_{i=1}^{k} u_{1,i} w_i - \sum_{i=1}^{k} u_{2,i} w_i = \sum_{i=1}^{k} (u_{1,i} - u_{2,i}) w_i$$

And, therefore, since w_i are linearly independent, $u_1 = u_2$.

DECODING of LINEAR CODES

Decoding problem: If a codeword: $x = x_1 \dots x_n$ is sent and the word $y = y_1 \dots y_n$ is received, then $e = y - x = e_1 \dots e_n$ is said to be the error vector. The decoder must decide, from y, which x was sent, or, equivalently, which error e occurred.

To describe main Decoding method some technicalities have to be introduced

Definition Suppose C is an $[n, k]$-code over $GF(q)$ and $u \in V(n, q)$. Then the set

$$u + C = \{u + x | x \in C\}$$

is called a coset (u-coset) of C in $V(n, q)$.

DECODING of LINEAR CODES

Decoding problem: If a codeword: $x = x_1 \dots x_n$ is sent and the word $y = y_1 \dots y_n$ is received, then $e = y - x = e_1 \dots e_n$ is said to be the error vector. The decoder must decide, from y, which x was sent, or, equivalently, which error e occurred.

To describe main Decoding method some technicalities have to be introduced

Definition Suppose C is an $[n, k]$-code over $GF(q)$ and $u \in V(n, q)$. Then the set

$$u + C = \{u + x | x \in C\}$$

is called a coset (u-coset) of C in $V(n, q)$.

Example Let $C = \{0000, 1011, 0101, 1110\}$
Cosets:
$0000 + C = C$,
$1000 + C = \{1000, 0011, 1101, 0110\}$,
$0100 + C = \{0100, 1111, 0001, 1010\} = 0001 + C$,
$0010 + C = \{0010, 1001, 0111, 1100\}$.
Are there some other cosets in this case?

DECODING of LINEAR CODES

Decoding problem: If a codeword: $x = x_1 \dots x_n$ is sent and the word $y = y_1 \dots y_n$ is received, then $e = y - x = e_1 \dots e_n$ is said to be the error vector. The decoder must decide, from y, which x was sent, or, equivalently, which error e occurred.

To describe main Decoding method some technicalities have to be introduced

Definition Suppose C is an $[n, k]$-code over $GF(q)$ and $u \in V(n, q)$. Then the set

$$u + C = \{u + x | x \in C\}$$

is called a coset (u-coset) of C in $V(n, q)$.

Example Let $C = \{0000, 1011, 0101, 1110\}$
Cosets:
$0000 + C = C$,
$1000 + C = \{1000, 0011, 1101, 0110\}$,
$0100 + C = \{0100, 1111, 0001, 1010\} = 0001 + C$,
$0010 + C = \{0010, 1001, 0111, 1100\}$.
Are there some other cosets in this case?
Theorem Suppose C is a linear $[n, k]$-code over $GF(q)$. Then

(a) every vector of $V(n, q)$ is in some coset of C,

(b) every coset contains exactly q^k elements,

(c) two cosets are either disjoint or identical.

NEAREST NEIGHBOUR DECODING SCHEME

Each vector having minimum weight in a coset is called a coset leader.

1. Design a (Slepian) standard array for an $[n, k]$-code C - that is a $q^{n-k} \times q^k$ array of the form:

codewords	coset leader	codeword 2	...	codeword 2^k
	coset leader	$+$	...	$+$
	...	$+$	$+$	$+$
	coset leader	$+$	...	$+$
	coset leader			

NEAREST NEIGHBOUR DECODING SCHEME

Each vector having minimum weight in a coset is called a coset leader.

1. Design a (Slepian) standard array for an $[n, k]$-code C - that is a $q^{n-k} \times q^k$ array of the form:

codewords	coset leader	codeword 2	...	codeword 2^k
	coset leader	$+$	...	$+$
	...	$+$	$+$	$+$
	coset leader	$+$	...	$+$
	coset leader			

Example

0000	1011	0101	1110
1000	0011	1101	0110
0100	1111	0001	1010
0010	1001	0111	1100

A word y is decoded as codeword of the first row of the column in which y occurs.
Error vectors which will be corrected are precisely coset leaders!
In practice, this decoding method is too slow and requires too much memory.

PROBABILITY of GOOD ERROR CORRECTION

What is the probability that a received word will be decoded correctly - that is as the codeword that was sent (for binary linear codes and binary symmetric channel)?

Probability of an error in the case of a given error vector of weight i is

$$p^i(1-p)^{n-i}.$$

Therefore, it holds.

Theorem Let C be a binary $[n, k]$-code, and for $i = 0, 1, \dots, n$ let α_i be the number of coset leaders of weight i. The probability $P_{corr}(C)$ that a received vector when decoded by means of a standard array is the codeword which was sent is given by

$$P_{corr}(C) = \sum_{i=0}^{n} \alpha_i p^i (1-p)^{n-i}.$$

PROBABILITY of GOOD ERROR CORRECTION

What is the probability that a received word will be decoded correctly - that is as the codeword that was sent (for binary linear codes and binary symmetric channel)?

Probability of an error in the case of a given error vector of weight i is

$$p^i(1-p)^{n-i}.$$

Therefore, it holds.

Theorem Let C be a binary $[n,k]$-code, and for $i = 0, 1, \ldots, n$ let α_i be the number of coset leaders of weight i. The probability $P_{corr}(C)$ that a received vector when decoded by means of a standard array is the codeword which was sent is given by

$$P_{corr}(C) = \sum_{i=0}^{n} \alpha_i p^i (1-p)^{n-i}.$$

Example For the $[4,2]$-code of the last example

$$\alpha_0 = 1, \alpha_1 = 3, \alpha_2 = \alpha_3 = \alpha_4 = 0.$$

Hence

$$P_{corr}(C) = (1-p)^4 + 3p(1-p)^3 = (1-p)^3(1+2p).$$

If $p = 0.01$, then $P_{corr} = 0.9897$

PROBABILITY of GOOD ERROR DETECTION

Suppose a binary linear code is used only for error detection.

The decoder will fail to detect errors which have occurred if the received word y is a codeword different from the codeword x which was sent, i. e. if the error vector $e = y - x$ is itself a non-zero codeword.

The probability $P_{undetect}(C)$ that an incorrect codeword is received is given by the following result.

Theorem Let C be a binary $[n, k]$-code and let A_i denote the number of codewords of C of weight i. Then, if C is used for error detection, the probability of an incorrect message being received is

$$P_{undetect}(C) = \sum_{i=0}^{n} A_i p^i (1-p)^{n-i}.$$

PROBABILITY of GOOD ERROR DETECTION

Suppose a binary linear code is used only for error detection.

The decoder will fail to detect errors which have occurred if the received word y is a codeword different from the codeword x which was sent, i. e. if the error vector $e = y - x$ is itself a non-zero codeword.

The probability $P_{undetect}(C)$ that an incorrect codeword is received is given by the following result.

Theorem Let C be a binary $[n, k]$-code and let A_i denote the number of codewords of C of weight i. Then, if C is used for error detection, the probability of an incorrect message being received is

$$P_{undetect}(C) = \sum_{i=0}^{n} A_i p^i (1-p)^{n-i}.$$

Example In the case of the $[4, 2]$ code from the last example

$$A_2 = 1 \ A_3 = 2$$
$$P_{undetect}(C) = p^2(1-p)^2 + 2p^3(1-p) = p^2 - p^4.$$

For $p = 0.01$

$$P_{undetect}(C) = 0.00009999.$$

DUAL CODES

Inner product of two vectors (words)

$$u = u_1 \dots u_n, \quad v = v_1 \dots v_n$$

in $V(n,q)$ is an element of $GF(q)$ defined (using modulo q operations) by

$$u \cdot v = u_1 v_1 + \dots + u_n v_n.$$

Example *In* $V(4,2) : 1001 \cdot 1001 = 0$

In $V(4,3) : 2001 \cdot 1210 = 2$

$1212 \cdot 2121 = 2$

If $u \cdot v = 0$ then words (vectors) u and v are called orthogonal.

Properties If $u, v, w \in V(n,q), \lambda, \mu \in GF(q)$, then
$u \cdot v = v \cdot u, (\lambda u + \mu v) \cdot w = \lambda(u \cdot w) + \mu(v \cdot w)$.

Given a linear $[n,k]$-code C, then the dual code of C, denoted by $C^\perp$, is defined by

$$C^\perp = \{v \in V(n,q) \,|\, v \cdot u = 0 \text{ for all } u \in C\}.$$

Lemma Suppose C is an $[n,k]$-code having a generator matrix G. Then for $v \in V(n,q)$

$$v \in C^\perp \Leftrightarrow vG^\top = 0,$$

where $G^\top$ denotes the transpose of the matrix G.
Proof Easy.

PARITE CHECKS versus ORTHOGONALITY

For understanding of the role the parity checks play for linear codes, it is important to understand relation between orthogonality and special parity checks.

If binary words x and y are orthogonal, then the word y has even number of ones (1's) in the positions determined by ones (1's) in the word x.

This implies that if words x and y are orthogonal, then x is a parity check word for y and y is a parity check word for x.

Exercise: Let the word

$$100001$$

be orthogonal to a set S of binary words of length 6. What can we say about the words in S?

EXAMPLE

For the $[n, 1]$-repetition code C, with the generator matrix

$$G = (1, 1, \dots, 1)$$

the dual code $C^{\perp}$ is $[n, n-1]$-code with the generator matrix $G^{\perp}$, described by

$$G^{\perp} = \begin{pmatrix} 1 & 1 & 0 & 0 & \dots & 0 \\ 1 & 0 & 1 & 0 & \dots & 0 \\ & \dots & & & & \\ 1 & 0 & 0 & 0 & \dots & 1 \end{pmatrix}$$

PARITY CHECK MATRICES I

Example If

$$C_5 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}, \text{ then } C_5^\perp = C_5.$$

If

$$C_6 = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}, \text{ then } C_6^\perp = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}.$$

Example If

$$C_5 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}, \text{ then } C_5^{\perp} = C_5.$$

If

$$C_6 = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}, \text{ then } C_6^{\perp} = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}.$$

Theorem Suppose C is a linear $[n, k]$-code over $GF(q)$, then the dual code $C^{\perp}$ is a linear $[n, n-k]$-code.

Definition A parity-check matrix H for an $[n, k]$-code C is a generator matrix of $C^{\perp}$.

Definition A parity-check matrix H for an $[n, k]$-code C is a generator matrix of $C^{\perp}$.

Theorem If H is parity-check matrix of C, then

$$C = \{x \in V(n, q) | xH^{\top} = 0\},$$

and therefore any linear code is completely specified by a parity-check matrix.

PARITY CHECK MATRICES

Definition A parity-check matrix H for an $[n, k]$-code C is a generator matrix of $C^{\perp}$.

Theorem If H is parity-check matrix of C, then

$$C = \{x \in V(n,q) | xH^{\top} = 0\},$$

and therefore any linear code is completely specified by a parity-check matrix.

Example Parity-check matrix for

$$C_5 \text{ is } \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

and for

$$C_6 \text{ is } \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}$$

The rows of a parity check matrix are parity checks on codewords. They say that certain linear combinations of elements of every codeword are zeros.

Theorem If $G = [I_k | A]$ is the standard form generator matrix of an $[n, k]$-code C, then a parity check matrix for C is $H = [-A^\top | I_{n-k}]$.

Example

$$\text{Generator matrix } G = \left| I_4 \begin{vmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{vmatrix} \right| \Rightarrow \text{parity check m. } H = \left| \begin{vmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{vmatrix} I_3 \right|$$

Definition Suppose H is a parity-check matrix of an $[n, k]$-code C. Then for any $y \in V(n, q)$ the following word is called the syndrome of y:

$$S(y) = yH^\top.$$

Lemma Two words have the same syndrome iff they are in the same coset.

Syndrom decoding Assume that a standard array of a code C is given and, in addition, let in the last two columns the syndrome for each coset be given.

$$\begin{array}{cccc|cccc|cccc|cccc|cc}
0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0
\end{array}$$

When a word y is received, compute $S(y) = yH^\top$, locate $S(y)$ in the "syndrome column", and then locate y in the same row and decode y as the codeword in the same column and in the first row.

KEY OBSERVATION for SYNDROM COMPUTATION

When preparing a "syndrome decoding" it is sufficient to store only two columns: one for **coset leaders** and one for **syndromes**.

Example

coset leaders	syndromes
l(z)	z
0000	00
1000	11
0100	01
0010	10

Decoding procedure

- **Step 1** Given y compute $S(y)$.
- **Step 2** Locate $z = S(y)$ in the syndrome column.
- **Step 3** Decode y as $y - l(z)$.

KEY OBSERVATION for SYNDROM COMPUTATION

When preparing a "syndrome decoding" it is sufficient to store only two columns: one for **coset leaders** and one for **syndromes**.

Example

coset leaders	syndromes
l(z)	z
0000	00
1000	11
0100	01
0010	10

Decoding procedure

- **Step 1** Given y compute $S(y)$.
- **Step 2** Locate $z = S(y)$ in the syndrome column.
- **Step 3** Decode y as $y - l(z)$.

Example If $y = 1111$, then $S(y) = 01$ and the above decoding procedure produces

$$1111–0100 = 1011.$$

Syndrom decoding is much faster than searching for a nearest codeword to a received word. However, for large codes it is still too inefficient to be practical.

In general, the problem of finding the nearest neighbour in a linear code is NP-complete. Fortunately, there are important linear codes with really efficient decoding.

HAMMING CODES

An important family of simple linear codes that are easy to encode and decode, are so-called Hamming codes.

Definition Let r be an integer and H be an $r \times (2^r - 1)$ matrix columns of which are all non-zero distinct words from $V(r, 2)$. The code having H as its parity-check matrix is called binary Hamming code and denoted by $Ham(r, 2)$.

Example

$$Ham(2,2) : H = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \Rightarrow G = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

$$Ham(3,2) = H = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \Rightarrow G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

HAMMING CODES

An important family of simple linear codes that are easy to encode and decode, are so-called Hamming codes.

Definition Let r be an integer and H be an $r \times (2^r - 1)$ matrix columns of which are all non-zero distinct words from $V(r, 2)$. The code having H as its parity-check matrix is called binary Hamming code and denoted by $Ham(r, 2)$.

Example

$$Ham(2,2) : H = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \Rightarrow G = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

$$Ham(3,2) = H = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \Rightarrow G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Theorem Hamming code $Ham(r, 2)$

- is $[2^r - 1, 2^r - 1 - r]$-code,
- has minimum distance 3,
- is a perfect code.

Properties of binary Hamming codes Coset leaders are precisely words of weight ≤ 1. The syndrome of the word $0 \ldots 010 \ldots 0$ with 1 in j-th position and 0 otherwise is the transpose of the j-th column of H.

HAMMING CODES - DECODING

Decoding algorithm for the case the columns of H are arranged in the order of increasing binary numbers the columns represent.

- **Step 1** Given y compute syndrome $S(y) = yH^\top$.
- **Step 2** If $S(y) = 0$, then y is assumed to be the codeword sent.
- **Step 3** If $S(y) \neq 0$, then assuming a single error, $S(y)$ gives the binary position of the error.

EXAMPLE

For the Hamming code given by the parity-check matrix

$$H = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

and the received word

$$y = 1101011,$$

we get syndrome

$$S(y) = 110$$

and therefore the error is in the sixth position.

Hamming code was discovered by Hamming (1950), Golay (1950).

It was conjectured for some time that Hamming codes and two so called Golay codes are the only non-trivial perfect codes.

Comment

Hamming codes were originally used to deal with errors in long-distance telephon calls.

ADVANTAGES of HAMMING CODES

Let a binary symmetric channel be used which with probability q correctly transfers a binary symbol.

If a 4-bit message is transmitted through such a channel, then correct transmission of the message occurs with probability q^4.

If Hamming $(7, 4, 3)$ code is used to transmit a 4-bit message, then probability of correct decoding is

$$q^7 + 7(1 - q)q^6.$$

In case $q = 0.9$ the probability of correct transmission is 0.6561 in the case no error correction is used and 0.8503 in the case Hamming code is used - an essential improvement.

IMPORTANT CODES

- Hamming $(7, 4, 3)$-code. It has 16 codewords of length 7. It can be used to send $2^7 = 128$ messages and can be used to correct 1 error.
- Golay $(23, 12, 7)$-code. It has 4 096 codewords. It can be used to transmit 8 388 608 messages and can correct 3 errors.
- Quadratic residue $(47, 24, 11)$-code. It has

16 777 216 codewords

and can be used to transmit

140 737 488 355 238 messages

and correct 5 errors.

- Hamming and Golay codes are the only non-trivial perfect codes.

GOLAY CODES - DESCRIPTION

Golay codes G_{24} and G_{23} were used by Voyager I and Voyager II to transmit color pictures of Jupiter and Saturn. Generation matrix for G_{24} has the form

$$G = \begin{pmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&1&1&1&0&1&1&1&0&0&0&1&0\\
0&1&0&0&0&0&0&0&0&0&0&0&1&0&1&1&0&1&1&1&0&0&0&1\\
0&0&1&0&0&0&0&0&0&0&0&0&1&1&0&1&1&0&1&1&1&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&1&0&1&0&1&1&0&1&1&1&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&1&0&1&1&0&1&1&1&0\\
0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&1&0&1&1&0&1&1&1\\
0&0&0&0&0&0&1&0&0&0&0&0&1&1&0&0&0&1&0&1&1&0&1&1\\
0&0&0&0&0&0&0&1&0&0&0&0&1&1&1&0&0&0&1&0&1&1&0&1\\
0&0&0&0&0&0&0&0&1&0&0&0&1&1&1&1&0&0&0&1&0&1&1&0\\
0&0&0&0&0&0&0&0&0&1&0&0&1&0&1&1&1&0&0&0&1&0&1&1\\
0&0&0&0&0&0&0&0&0&0&1&0&1&1&0&1&1&1&0&0&0&1&0&1\\
0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&0&1&1&1&0&0&0&1&0
\end{pmatrix}$$

G_{24} is $(24, 12, 8)$-code and the weights of all codewords are multiples of 4. G_{23} is obtained from G_{24} by deleting last symbols of each codeword of G_{24}. G_{23} is $(23, 12, 7)$-code.

GOLAY CODES - CONSTRUCTION

Matrix G for Golay code G_{24} has actually a simple and regular construction.

The first 12 columns are formed by a unitary matrix I_{12}, next column has all 1's.

Rows of the last 11 columns are cyclic permutations of the first row which has 1 at those positions that are squares modulo 11, that is

$$0, 1, 3, 4, 5, 9.$$

REED-MULLER CODES

Reed-Muller codes form a family of codes defined recursively with interesting properties and easy decoding.

If D_1 is a binary $[n, k_1, d_1]$-code and D_2 is a binary $[n, k_2, d_2]$-code, a binary code C of length $2n$ is defined as follows $C = \{u|u + v, \textit{where } u \in D_1, v \in D_2\}$.

Lemma C is $[2n, k_1 + k_2, min\{2d_1, d_2\}]$-code and if G_i is a generator matrix for D_i, $i = 1, 2$, then $\begin{bmatrix} G_1 & G_1 \\ 0 & G_2 \end{bmatrix}$ is a generator matrix for C.

Reed-Muller codes $R(r, m)$, with $0 \leq r \leq m$ are binary codes of length $n = 2^m . R(m, m)$ is the whole set of words of length n, $R(0, m)$ is the repetition code.

If $0 < r < m$, then $R(r + 1, m + 1)$ is obtained from codes $R(r + 1, m)$ and $R(r, m)$ by the above construction.

Theorem The dimension of $R(r, m)$ equals $1 + \binom{m}{1} + \ldots + \binom{m}{r}$. The minimum weight of $R(r, m)$ equals 2^{m-r}. Codes $R(m - r - 1, m)$ and $R(r, m)$ are dual codes.

SINGLETON BOUND

Singleton bound: Let C be a q-ary (n, M, d)-code.
Then

$$M \leq q^{n-d+1}.$$

Proof Take some $d-1$ coordinates and project all codewords to the resulting coordinates.

The resulting codewords are all different and therefore M cannot be larger than the number of q-ary words of length $n-d-1$.

Codes for which $M = q^{n-d+1}$ are called MDS-codes (Maximum Distance Separable).

Corollary: If C is a q-ary linear $[n, k, d]$-code, then

$$k + d \leq n + 1.$$

SHORTENING and PUNCTURING of LINEAR CODES

Let C be a q-ary linear $[n, k, d]$-code. Let

$D = \{(x_1, \dots, x_{n-1}) | (x_1, \dots, x_{n-1}, 0) \in C\}$. then D is a linear code - a shortening of the code C.

If $d > 1$, then D is a linear $[n-1, k, d^*]$-code or $[n-1, k, d-1]$-code a shortening of the code C.

Corollary: If there is a q-ary $[n, k, d]$-code, then shortening yields a q-ary $[n-1, k-1, d]$-code.

Let C be a q-ary $[n, k, d]$-code. Let

$$E = \{(x_1, \dots, x_{n-1}) | (x_1, \dots, x_{n-1}, x) \in C, \textit{for some } x \leq q\},$$

then E is a linear code - a puncturing of the code C.

If $d > 1$, then E is an $[n-1, k, d^*]$ code where $d^* = d-1$ if C has a minimum weight codeword with wit non-zero llast coordinate and $D^* = d$ otherwise.

when $d = 1$, then E is an $[n-1, k, 1]$ code, if C has no codeword of weight 1 whose nonzero entry is in last coordinate; otherwise, if $k > 1$, then E s an $[n-1, k-1, d^*]$

REED-SOLOMON CODES

An important example of MDS-codes are q-ary Reed-Solomon codes RSC(k, q), for $k \leq q$.

They are codes generator matrix of which has rows labelled by polynomials X^i, $0 \leq i \leq k-1$, columns by elements $0, 1, \ldots, q-1$ and the element in a row labelled by a polynomial p and in a column labelled by an element u is $p(u)$.

RSC(k, q) code is $[q, k, q-k+1]$ code.

Example Generator matrix for RSC$(3, 5)$ code is

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 1 & 4 & 4 & 1 \end{bmatrix}$$

Interesting property of Reed-Solomon codes:

$$\text{RSC}(k, q)^{\perp} = \text{RSC}(q-k, q).$$

Reed-Solomon codes are used in digital television, satellite communication, wireless communication, barcodes, compact discs, DVD,... They are very good to correct burst errors - such as ones caused by solar energy.

SOCCER GAMES BETTING SYSTEM

Ternary Golay code with parameters (11, 729, 5) can be used to bet for results of 11 soccer games with potential outcomes 1 (if home team wins), 2 (if guests win) and 3 (in case of a draw).

If 729 bets are made, then at least one bet has at least 9 results correctly guessed.

In case one has to bet for 13 games, then one can usually have two games with pretty sure outcomes and for the rest one can use the above ternary Golay code.

LDPC (Low-Density Parity Check) - CODES

A LDPC code is a binary linear code whose parity check matrix is very sparse - it contains only very few 1's.

A linear $[n, k]$ code is a regular $[n, k, r, c]$ LDPC code if $r << n, c << n - k$ and its parity-check matrix has exactly r 1's in each row and exactly c 1's in each column.

In the last years LDPC codes are replacing in many important applications other types of codes for the following reasons:

1. LDPC codes are in principle also very good channel codes, so called **Shannon capacity approaching codes**, they allow the noise threshold to be set arbitrarily close to the theoretical maximum - to Shannon limit - for symmetric channel.
2. Good LDPC codes can be decoded in time linear to their block length using special (for example "iterative belief propagation") approximation techniques.
3. Some LDPC codes are well suited for implementations that make heavy use of parallelism.

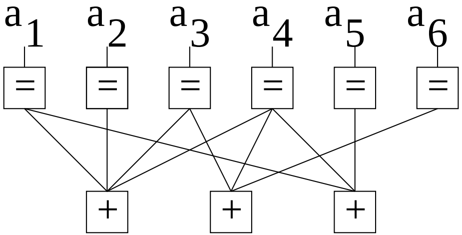
Parity-check matrices for LDPC codes are often (pseudo)-randomly generated, subject to sparsity constrains. Such LDPC codes are proven to be good with a high probability.

DISCOVERY and APPLICATION of LDPC CODES

LDPC codes were discovered in 1960 by R.C. Gallager in his PhD thesis, but ignored till 1996 when linear time decoding methods were discovered for some of them.

LDPC codes are used for: deep space communication; digital video broadcasting; 10GBase-T Ethernet, which sends data at 10 gigabits per second over Twisted-pair cables; Wi-Fi standard,....

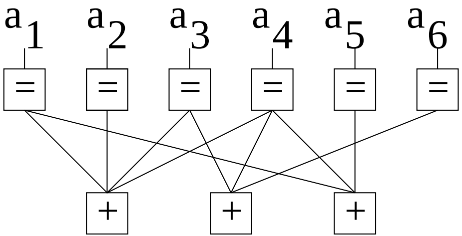
An $[n, k]$ LDPC code can be represented by a bipartite graph between a set of n top "variable-nodes (v-nodes)" and a set of bottom $(n-k)$ "constrain nodes (c-nodes)".



The corresponding parity check matrix has $n-k$ rows and n columns and i-th column has 1 in the j-th row exactly in case if i-th v-node is connected to j-th c-node.

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

Valid codewords for the LDPC-code with Tanner graph



with parity check matrix

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

have to satisfy constrains

$$a_1 + a_2 + a_3 + a_4 = 0$$

$$a_3 + a_4 + a_6 = 0$$

$$a_1 + a_4 + a_5 = 0$$

APPENDIX

COMMENTS

- $GF(q)$ for a prime q is the set $\{0, 1, \ldots, q-1\}$ with operations $+$ and $\cdot$ modulo q.

Part III

Cyclic codes and channel codes

CHAPTER 3: CYCLIC CODES and CHANNEL CODES

Cyclic codes are special linear codes of large interest and importance because

- They posses a rich algebraic structure that can be utilized in a variety of ways.
- They have extremely concise specifications.
- Their encodings can be efficiently implemented using simple shift registers.
- Many of the practically very important codes are cyclic.

Channel codes are used to encode streams of data (bits). Some of them, as Turbo codes, reach theoretical Shannon bound concerning efficiency, and are currently used often.

IMPORTANT NOTE

In order to specify a binary code with 2^k codewords of length n one may need to write down

$$2^k$$

codewords of length n.

IMPORTANT NOTE

In order to specify a binary code with 2^k codewords of length n one may need to write down

$$2^k$$

codewords of length n.

In order to specify a linear binary code of the dimension k with 2^k codewords of length n it is sufficient to write down

$$k$$

codewords of length n.

IMPORTANT NOTE

In order to specify a binary code with 2^k codewords of length n one may need to write down

$$2^k$$

codewords of length n.

In order to specify a linear binary code of the dimension k with 2^k codewords of length n it is sufficient to write down

$$k$$

codewords of length n.

In order to specify a binary cyclic code with 2^k codewords of length n it is sufficient to write down

$$1$$

codeword of length n.

Definition A code C is cyclic if

(i) C is a linear code;

(ii) any cyclic shift of a codeword is also a codeword, i.e. whenever $a_0, \dots a_{n-1} \in C$, then also $a_{n-1}a_0 \dots a_{n-2} \in C$ and $a_1 a_2 \dots a_{n-1} a_0 \in C$.

BASIC DEFINITION AND EXAMPLES

Definition A code C is cyclic if

(i) C is a linear code;

(ii) any cyclic shift of a codeword is also a codeword, i.e. whenever $a_0, \ldots a_{n-1} \in C$, then also $a_{n-1}a_0 \ldots a_{n-2} \in C$ and $a_1 a_2 \ldots a_{n-1} a_0 \in C$.

Example

(i) Code $C = \{000, 101, 011, 110\}$ is cyclic.

(ii) Hamming code $Ham(3,2)$: with the generator matrix

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

is equivalent to a cyclic code.

(iii) The binary linear code $\{0000, 1001, 0110, 1111\}$ is not cyclic, but it is equivalent to a cyclic code.

(iv) Is Hamming code $Ham(2,3)$ with the generator matrix

$$\begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \end{bmatrix}$$

(a) cyclic?

(b) or at least equivalent to a cyclic code?

FREQUENCY of CYCLIC CODES

Comparing with linear codes, cyclic codes are quite scarce. For example, there are 11 811 linear [7,3] binary codes, but only two of them are cyclic.

Trivial cyclic codes. For any field F and any integer $n \geq 3$ there are always the following cyclic codes of length n over F:

- No-information code - code consisting of just one all-zero codeword.
- Repetition code - code consisting of all codewords (a, a, ...,a) for $a \in F$.
- Single-parity-check code - code consisting of all codewords with parity 0.
- No-parity code - code consisting of all codewords of length n

For some cases, for example for $n = 19$ and $F = GF(2)$, the above four trivial cyclic codes are the only cyclic codes.

EXAMPLE of a CYCLIC CODE

The code with the generator matrix

$$G = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

has, in addition to the codeword 0000000, the following codewords

$$\begin{aligned} c_1 &= 1011100 \\ c_1 + c_2 &= 1110010 \end{aligned} \qquad \begin{aligned} c_2 &= 0101110 \\ c_1 + c_3 &= 1001011 \\ c_1 + c_2 + c_3 &= 1100101 \end{aligned} \qquad \begin{aligned} c_3 &= 0010111 \\ c_2 + c_3 &= 0111001 \end{aligned}$$

and it is cyclic because the right shifts have the following impacts

$$\begin{aligned} c_1 &\longrightarrow c_2, \\ c_1 + c_2 &\longrightarrow c_2 + c_3, \end{aligned} \qquad \begin{aligned} c_2 &\longrightarrow c_3, \\ c_1 + c_3 &\longrightarrow c_1 + c_2 + c_3, \\ c_1 + c_2 + c_3 &\longrightarrow c_1 + c_2 \end{aligned} \qquad \begin{aligned} c_3 &\longrightarrow c_1 + c_3 \\ c_2 + c_3 &\longrightarrow c_1 \end{aligned}$$

POLYNOMIALS over GF(q)

A codeword of a cyclic code is usually denoted

$$a_0 a_1 \ldots a_{n-1}$$

and to each such a codeword the polynomial

$$a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1}$$

will be associated.

POLYNOMIALS over GF(q)

A codeword of a cyclic code is usually denoted

$$a_0 a_1 \dots a_{n-1}$$

and to each such a codeword the polynomial

$$a_0 + a_1 x + a_2 x^2 + \dots + a_{n-1} x^{n-1}$$

will be associated.

NOTATION: $F_q[x]$ denotes the set of all polynomials over $GF(q)$.

$deg(f(x))$ = the largest m such that x^m has a non-zero coefficient in $f(x)$.

POLYNOMIALS over GF(q)

A codeword of a cyclic code is usually denoted

$$a_0 a_1 \dots a_{n-1}$$

and to each such a codeword the polynomial

$$a_0 + a_1 x + a_2 x^2 + \dots + a_{n-1} x^{n-1}$$

will be associated.

NOTATION: $F_q[x]$ denotes the set of all polynomials over $GF(q)$.

$deg(f(x))$ = the largest m such that x^m has a non-zero coefficient in $f(x)$.

Multiplication of polynomials If $f(x)$, $g(x) \in Fq[x]$, then

$$deg(f(x)g(x)) = deg(f(x)) + deg(g(x)).$$

POLYNOMIALS over GF(q)

A codeword of a cyclic code is usually denoted

$$a_0 a_1 \dots a_{n-1}$$

and to each such a codeword the polynomial

$$a_0 + a_1 x + a_2 x^2 + \dots + a_{n-1} x^{n-1}$$

will be associated.

NOTATION: $F_q[x]$ denotes the set of all polynomials over $GF(q)$.

$deg(f(x))$ = the largest m such that x^m has a non-zero coefficient in $f(x)$.

Multiplication of polynomials If $f(x)$, $g(x) \in Fq[x]$, then

$$deg(f(x)g(x)) = deg(f(x)) + deg(g(x)).$$

Division of polynomials For every pair of polynomials $a(x)$, $b(x) \neq 0$ in $F_q[x]$ there exists a unique pair of polynomials $q(x)$, $r(x)$ in $F_q[x]$ such that

$$a(x) = q(x)b(x) + r(x), deg(r(x)) < deg(b(x)).$$

Example Divide $x^3 + x + 1$ by $x^2 + x + 1$ in $F_2[x]$.

POLYNOMIALS over GF(q)

A codeword of a cyclic code is usually denoted

$$a_0 a_1 \dots a_{n-1}$$

and to each such a codeword the polynomial

$$a_0 + a_1 x + a_2 x^2 + \dots + a_{n-1} x^{n-1}$$

will be associated.

NOTATION: $F_q[x]$ denotes the set of all polynomials over $GF(q)$.

$deg(f(x))$ = the largest m such that x^m has a non-zero coefficient in $f(x)$.

Multiplication of polynomials If $f(x)$, $g(x) \in Fq[x]$, then

$$deg(f(x)g(x)) = deg(f(x)) + deg(g(x)).$$

Division of polynomials For every pair of polynomials $a(x)$, $b(x) \neq 0$ in $F_q[x]$ there exists a unique pair of polynomials $q(x)$, $r(x)$ in $F_q[x]$ such that

$$a(x) = q(x)b(x) + r(x), deg(r(x)) < deg(b(x)).$$

Example Divide $x^3 + x + 1$ by $x^2 + x + 1$ in $F_2[x]$.
Definition Let $f(x)$ be a fixed polynomial in $F_q[x]$. Two polynomials $g(x)$, $h(x)$ are said to be congruent modulo $f(x)$, notation

$$g(x) \equiv h(x) (\text{mod } f(x)),$$

if $g(x) - h(x)$ is divisible by $f(x)$.

For any polynomial $f(x)$, the set of all polynomials in $F_q[x]$ of degree less than $deg(f(x))$, with addition and multiplication modulo $f(x)$, forms a **ring denoted** $F_q[x]/f(x)$.

Example Calculate $(x+1)^2$ in $F_2[x]/(x^2+x+1)$. It holds

$$(x+1)^2 = x^2+2x+1 \equiv x^2+1 \equiv x \ (\text{mod } x^2+x+1).$$

How many elements has $F_q[x]/f(x)$?

Result $|F_q[x]/f(x)| = q^{deg(f(x))}$.

For any polynomial $f(x)$, the set of all polynomials in $F_q[x]$ of degree less than $deg(f(x))$, with addition and multiplication modulo $f(x)$, forms a **ring denoted** $F_q[x]/f(x)$.

Example Calculate $(x+1)^2$ in $F_2[x]/(x^2+x+1)$. It holds

$$(x+1)^2 = x^2+2x+1 \equiv x^2+1 \equiv x \pmod{x^2+x+1}.$$

How many elements has $F_q[x]/f(x)$?

Result $|F_q[x]/f(x)| = q^{deg(f(x))}$.

Example Addition and multiplication tables for $F_2[x]/(x^2+x+1)$

+	0	1	x	1+x
0	0	1	x	1+x
1	1	0	1+x	x
x	x	1+x	0	1
1+x	1+x	x	1	0

•	0	1	x	1+x
0	0	0	0	0
1	0	1	x	1+x
x	0	x	1+x	1
1+x	0	1+x	1	x

RINGS of POLYNOMIALS

For any polynomial $f(x)$, the set of all polynomials in $F_q[x]$ of degree less than $deg(f(x))$, with addition and multiplication modulo $f(x)$, forms a **ring denoted** $F_q[x]/f(x)$.

Example Calculate $(x+1)^2$ in $F_2[x]/(x^2+x+1)$. It holds

$$(x+1)^2 = x^2 + 2x + 1 \equiv x^2 + 1 \equiv x \pmod{x^2+x+1}.$$

How many elements has $F_q[x]/f(x)$?

Result $|F_q[x]/f(x)| = q^{deg(f(x))}$.

Example Addition and multiplication tables for $F_2[x]/(x^2+x+1)$

+	0	1	x	1+x
0	0	1	x	1+x
1	1	0	1+x	x
x	x	1+x	0	1
1+x	1+x	x	1	0

•	0	1	x	1+x
0	0	0	0	0
1	0	1	x	1+x
x	0	x	1+x	1
1+x	0	1+x	1	x

Definition A polynomial $f(x)$ in $F_q[x]$ is said to be reducible if $f(x) = a(x)b(x)$, where $a(x), b(x) \in F_q[x]$ and

$$deg(a(x)) < deg(f(x)), \qquad deg(b(x)) < deg(f(x)).$$

If $f(x)$ is not reducible, then it is said to be irreducible in $F_q[x]$.
Theorem The ring $F_q[x]/f(x)$ is a field if $f(x)$ is irreducible in $F_q[x]$.

Computation modulo $x^n - 1$ in the field $R_n = F_q[x]/(x^n - 1)$

Since $x^n \equiv 1 (\bmod\ (x^n - 1))$ we can compute $f(x) \bmod (x^n - 1)$ by replacing, in $f(x)$, $x^n by 1$, x^{n+1} by x, x^{n+2} by x^2, x^{n+3} by x^3, ...

FIELD $R_n, R_n = F_q[x]/(x^n - 1)$

Computation modulo $x^n - 1$ in the field $R_n = F_q[x]/(x^n - 1)$

Since $x^n \equiv 1 (\text{mod } (x^n - 1))$ we can compute $f(x) \text{ mod } (x^n - 1)$ by replacing, in $f(x)$, $x^n by 1$, x^{n+1} by x, x^{n+2} by x^2, x^{n+3} by x^3, ...

Replacement of a word

$$w = a_0 a_1 \ldots a_{n-1}$$

by a polynomial

$$p(w) = a_0 + a_1 x + \ldots + a_{n-1} x^{n-1}$$

is of large importance because

multiplication of $p(w)$ by x in R_n corresponds to a single cyclic shift of w

$$x(a_0 + a_1 x + \ldots a_{n-1} x^{n-1}) = a_{n-1} + a_0 x + a_1 x^2 + \ldots + a_{n-2} x^{n-1}$$

Theorem A code C is cyclic if and only if it satisfies two conditions

(i) $a(x), b(x) \in C \Rightarrow a(x) + b(x) \in C$

(ii) $a(x) \in C, r(x) \in R_n \Rightarrow r(x)a(x) \in C$

Theorem A code C is cyclic if and only if it satisfies two conditions

(i) $a(x), b(x) \in C \Rightarrow a(x) + b(x) \in C$

(ii) $a(x) \in C, r(x) \in R_n \Rightarrow r(x)a(x) \in C$

Proof

(1) Let C be a cyclic code. C is linear $\Rightarrow$

(i) holds.

(ii)

$$Let\ a(x) \in C, r(x) = r_0 + r_1 x + \ldots + r_{n-1}x^{n-1}$$

$$r(x)a(x) = r_0 a(x) + r_1 x a(x) + \ldots + r_{n-1}x^{n-1}a(x)$$

is in C by (i) because summands are cyclic shifts of $a(x)$.

(2) Let (i) and (ii) hold

- Taking $r(x)$ to be a scalar the conditions imply linearity of C.
- Taking $r(x) = x$ the conditions imply cyclicity of C.

CONSTRUCTION of CYCLIC CODES

Notation For any $f(x) \in R_n$, we can define

$$\langle f(x) \rangle = \{r(x)f(x) \mid r(x) \in R_n\}$$

(with multiplication modulo $x^n - 1$) a set of polynomials - a code.

CONSTRUCTION of CYCLIC CODES

Notation For any $f(x) \in R_n$, we can define

$$\langle f(x) \rangle = \{ r(x) f(x) \mid r(x) \in R_n \}$$

(with multiplication modulo $x^n - 1$) a set of polynomials - a code.

Theorem For any $f(x) \in R_n$, the set $\langle f(x) \rangle$ is a cyclic code (generated by f).

Proof We check conditions (i) and (ii) of the previous theorem.

(i) If $a(x)f(x) \in \langle f(x) \rangle$ and also $b(x)f(x) \in \langle f(x) \rangle$, then

$$a(x)f(x) + b(x)f(x) = (a(x) + b(x))f(x) \in \langle f(x) \rangle$$

(ii) If $a(x)f(x) \in \langle f(x) \rangle$, $r(x) \in R_n$, then

$$r(x)(a(x)f(x)) = (r(x)a(x))f(x) \in \langle f(x) \rangle$$

CONSTRUCTION of CYCLIC CODES

Notation For any $f(x) \in R_n$, we can define

$$\langle f(x) \rangle = \{r(x)f(x) \mid r(x) \in R_n\}$$

(with multiplication modulo $x^n - 1$) a set of polynomials - a code.

Theorem For any $f(x) \in R_n$, the set $\langle f(x) \rangle$ is a cyclic code (generated by f).

Proof We check conditions (i) and (ii) of the previous theorem.

(i) If $a(x)f(x) \in \langle f(x) \rangle$ and also $b(x)f(x) \in \langle f(x) \rangle$, then

$$a(x)f(x) + b(x)f(x) = (a(x) + b(x))f(x) \in \langle f(x) \rangle$$

(ii) If $a(x)f(x) \in \langle f(x) \rangle$, $r(x) \in R_n$, then

$$r(x)(a(x)f(x)) = (r(x)a(x))f(x) \in \langle f(x) \rangle$$

Example let $C = \langle 1 + x^2 \rangle$, $n = 3$, $q = 2$.
In order to determine C we have to compute $r(x)(1 + x^2)$ for all $r(x) \in R_3$.

$$R_3 = \{0, 1, x, 1 + x, x^2, 1 + x^2, x + x^2, 1 + x + x^2\}.$$

Result

$$C = \{0, 1 + x, 1 + x^2, x + x^2\}$$
$$C = \{000, 011, 101, 110\}$$

CHARACTERIZATION THEOREM for CYCLIC CODES

We show that all cyclic codes C have the form $C = \langle f(x) \rangle$ for some $f(x) \in R_n$.

Theorem Let C be a non-zero cyclic code in R_n. Then

- there exists a unique monic polynomial $g(x)$ of the smallest degree such that
- $C = \langle g(x) \rangle$
- $g(x)$ is a factor of $x^n - 1$.

CHARACTERIZATION THEOREM for CYCLIC CODES

We show that all cyclic codes C have the form $C = \langle f(x) \rangle$ for some $f(x) \in R_n$.

Theorem Let C be a non-zero cyclic code in R_n. Then

- there exists a unique monic polynomial $g(x)$ of the smallest degree such that
- $C = \langle g(x) \rangle$
- $g(x)$ is a factor of $x^n - 1$.

Proof

(i) Suppose $g(x)$ and $h(x)$ are two monic polynomials in C of the smallest degree. Then the polynomial $g(x) - h(x) \in C$ and it has a smaller degree and a multiplication by a scalar makes out of it a monic polynomial. If $g(x) \neq h(x)$ we get a contradiction.

(ii) Suppose $a(x) \in C$.

Then

$$a(x) = q(x)g(x) + r(x), \qquad (deg\ r(x) < deg\ g(x)).$$

and

$$r(x) = a(x) - q(x)g(x) \in C.$$

By minimality

$$r(x) = 0$$

and therefore $a(x) \in \langle g(x) \rangle$.

CHARACTERIZATION THEOREM for CYCLIC CODES - continuation

(iii) Clearly,

$$x^n - 1 = q(x)g(x) + r(x) \quad \text{with} \quad deg\ r(x) < deg\ g(x)$$

and therefore

$$r(x) \equiv -q(x)g(x) (\text{mod } x^n - 1) \quad \text{and}$$
$$r(x) \in C \Rightarrow r(x) = 0 \Rightarrow g(x) \quad \text{is a factor of } x^n - 1.$$

(iii) Clearly,

$$x^n - 1 = q(x)g(x) + r(x) \quad \text{with} \quad deg\ r(x) < deg\ g(x)$$

and therefore

$$r(x) \equiv -q(x)g(x) (\text{mod } x^n - 1) \quad \text{and}$$
$$r(x) \in C \Rightarrow r(x) = 0 \Rightarrow g(x) \quad \text{is a factor of } x^n - 1.$$

GENERATOR POLYNOMIALS

Definition If

$$C = \langle g(x) \rangle,$$

holds for a cyclic code C, then g is called the generator polynomial for the code C.

HOW TO DESIGN CYCLIC CODES?

The last claim of the previous theorem gives a recipe to get all cyclic codes of the given length n in GF(q).

Indeed, all we need to do is to find all factors (in GF(q)) of

$$x^n - 1.$$

Problem: Find all binary cyclic codes of length 3.

Solution: Since

$$x^3 - 1 = \underbrace{(x-1)(x^2+x+1)}_{\text{both factors are irreducible in GF(2)}}$$

we have the following generator polynomials and codes.

Generator polynomials	Code in R_3	Code in $V(3,2)$
1	R_3	$V(3,2)$
$x+1$	$\{0, 1+x, x+x^2, 1+x^2\}$	$\{000, 110, 011, 101\}$
x^2+x+1	$\{0, 1+x+x^2\}$	$\{000, 111\}$
$x^3-1\ (=0)$	$\{0\}$	$\{000\}$

DESIGN of GENERATOR MATRICES for CYCLIC CODES

Theorem Suppose C is a cyclic code of codewords of length n with the generator polynomial

$$g(x) = g_0 + g_1 x + \ldots + g_r x^r.$$

Then $dim\ (C) = n - r$ and a generator matrix G_1 for C is

$$G_1 = \begin{pmatrix} g_0 & g_1 & g_2 & \ldots & g_r & 0 & 0 & 0 & \ldots & 0 \\ 0 & g_0 & g_1 & g_2 & \ldots & g_r & 0 & 0 & \ldots & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & \ldots & g_r & 0 & \ldots & 0 \\ \ldots & \ldots & & & & & & & & \ldots \\ 0 & 0 & \ldots & 0 & 0 & \ldots & 0 & g_0 & \ldots & g_r \end{pmatrix}$$

DESIGN of GENERATOR MATRICES for CYCLIC CODES

Theorem Suppose C is a cyclic code of codewords of length n with the generator polynomial

$$g(x) = g_0 + g_1 x + \ldots + g_r x^r.$$

Then $dim\ (C) = n - r$ and a generator matrix G_1 for C is

$$G_1 = \begin{pmatrix} g_0 & g_1 & g_2 & \ldots & g_r & 0 & 0 & 0 & \ldots & 0 \\ 0 & g_0 & g_1 & g_2 & \ldots & g_r & 0 & 0 & \ldots & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & \ldots & g_r & 0 & \ldots & 0 \\ \ldots & \ldots & & & & & & & & \ldots \\ 0 & 0 & \ldots & 0 & 0 & \ldots & 0 & g_0 & \ldots & g_r \end{pmatrix}$$

Proof

(i) All rows of G1 are linearly independent.

(ii) The $n-r$ rows of G represent codewords

$$g(x), xg(x), x^2 g(x), \ldots, x^{n-r-1} g(x) \quad (*)$$

(iii) It remains to show that every codeword in C can be expressed as a linear combination of vectors from (*).

Indeed, if $a(x) \in C$, then

$$a(x) = q(x)g(x).$$

Since $deg\ a(x) < n$ we have $deg\ q(x) < n - r$.
Hence

$$\begin{aligned} q(x)g(x) &= (q_0 + q_1 x + \ldots + q_{n-r-1} x^{n-r-1})g(x) \\ &= q_0 g(x) + q_1 x g(x) + \ldots + q_{n-r-1} x^{n-r-1} g(x). \end{aligned}$$

EXAMPLE

The task is to determine all ternary codes of length 4 and generators for them.
Factorization of $x^4 - 1$ over $GF(3)$ has the form

$$x^4 - 1 = (x-1)(x^3 + x^2 + x + 1) = (x-1)(x+1)(x^2+1)$$

Therefore there are $2^3 = 8$ divisors of $x^4 - 1$ and each generates a cyclic code.

Generator polynomial	Gener
1	
$x - 1$	$\begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}$
$x + 1$	$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$
$x^2 + 1$	$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$
$(x-1)(x+1) = x^2 - 1$	$\begin{bmatrix} -1 \\ 0 \end{bmatrix}$
$(x-1)(x^2+1) = x^3 - x^2 + x - 1$	$\begin{bmatrix} -1 \end{bmatrix}$
$(x+1)(x^2+1)$	$\begin{bmatrix} 1 \end{bmatrix}$

COMMENT

On the previous slide "generator polynomials" $x-1$, x^2-1 and x^3-x^2+x+1 are formally not in R_n becasue only allowable coefficients are $0, 1, 2$.

A good practice is, however, to use also coefficients -2, and -1 as ones that are equal, modulo 3, to 1 nd 2 and they can be replace in such a way also in matrices to be fully correct formally.

CHECK POLYNOMIALS and PARITY CHECK MATRICES for CYCLIC CODES

Let C be a cyclic $[n, k]$-code with the generator polynomial $g(x)$ (of degree $n-k$). By the last theorem $g(x)$ is a factor of $x^n - 1$. Hence

$$x^n - 1 = g(x)h(x)$$

for some $h(x)$ of degree k. ($h(x)$ is called the check polynomial of C.)

Theorem Let C be a cyclic code in R_n with a generator polynomial $g(x)$ and a check polynomial $h(x)$. Then an $c(x) \in R_n$ is a codeword of C if and only if $c(x)h(x) \equiv 0$ –(this and next congruences are all modulo $x^n - 1$).

CHECK POLYNOMIALS and PARITY CHECK MATRICES for CYCLIC CODES

Let C be a cyclic $[n,k]$-code with the generator polynomial $g(x)$ (of degree $n-k$). By the last theorem $g(x)$ is a factor of $x^n - 1$. Hence

$$x^n - 1 = g(x)h(x)$$

for some $h(x)$ of degree k. ($h(x)$ is called the check polynomial of C.)

Theorem Let C be a cyclic code in R_n with a generator polynomial $g(x)$ and a check polynomial $h(x)$. Then an $c(x) \in R_n$ is a codeword of C if and only if $c(x)h(x) \equiv 0$ –(this and next congruences are all modulo $x^n - 1$).

Proof Note, that $g(x)h(x) = x^n - 1 \equiv 0$

(i) $c(x) \in C \Rightarrow c(x) = a(x)g(x)$ for some $a(x) \in R_n$

$$\Rightarrow c(x)h(x) = a(x)\underbrace{g(x)h(x)}_{\equiv 0} \equiv 0.$$

(ii) $c(x)h(x) \equiv 0$

$$c(x) = q(x)g(x) + r(x), deg\ r(x) < n-k = deg\ g(x)$$
$$c(x)h(x) \equiv 0 \Rightarrow r(x)h(x) \equiv 0 \pmod{x^n - 1}$$

Since $deg\ (r(x)h(x)) < n-k+k = n$, we have $r(x)h(x) = 0$ in $F[x]$ and therefore

$$r(x) = 0 \Rightarrow c(x) = q(x)g(x) \in C.$$

POLYNOMIAL REPRESENTATION of DUAL CODES

Since $dim\ (\langle h(x)\rangle) = n - k = dim(C^{\perp})$ we might easily be fooled to think that the check polynomial $h(x)$ of the code C generates the dual code $C^{\perp}$.

Reality is "slightly different":

Theorem Suppose C is a cyclic $[n, k]$-code with the check polynomial

$$h(x) = h_0 + h_1 x + \ldots + h_k x^k,$$

then

(i) a parity-check matrix for C is

$$H = \begin{pmatrix} h_k & h_{k-1} & \ldots & h_0 & 0 & \ldots & 0 \\ 0 & h_k & \ldots & h_1 & h_0 & \ldots & 0 \\ \ldots & \ldots & & & & & \\ 0 & 0 & \ldots & 0 & h_k & \ldots & h_0 \end{pmatrix}$$

(ii) $C^{\perp}$ is the cyclic code generated by the polynomial

$$\overline{h}(x) = h_k + h_{k-1}x + \ldots + h_0 x^k$$

i.e. the reciprocal polynomial of $h(x)$.

POLYNOMIAL REPRESENTATION of DUAL CODES

Proof A polynomial $c(x) = c_0 + c_1x + \ldots + c_{n-1}x^{n-1}$ represents a code from C if $c(x)h(x) = 0$. For $c(x)h(x)$ to be 0 the coefficients at $x^k, \ldots, x^{n-1}$ must be zero, i.e.

$$\begin{aligned}
c_0h_k + c_1h_{k-1} + \ldots + c_kh_0 &= 0 \\
c_1h_k + c_2h_{k-1} + \ldots + c_{k+1}h_0 &= 0 \\
&\ldots \\
c_{n-k-1}h_k + c_{n-k}h_{k-1} + \ldots + c_{n-1}h_0 &= 0
\end{aligned}$$

Therefore, any codeword $c_0c_1 \ldots c_{n-1} \in C$ is orthogonal to the word $h_kh_{k-1} \ldots h_000 \ldots 0$ and to its cyclic shifts.

Rows of the matrix H are therefore in $C^\perp$. Moreover, since $h_k = 1$, these row vectors are linearly independent. Their number is $n - k = dim\ (C^\perp)$. Hence H is a generator matrix for $C^\perp$, i.e. a parity-check matrix for C.

In order to show that $C^\perp$ is a cyclic code generated by the polynomial

$$\overline{h}(x) = h_k + h_{k-1}x + \ldots + h_0x^k$$

it is sufficient to show that $\overline{h}(x)$ is a factor of $x^n - 1$.

Observe that $\overline{h}(x) = x^kh(x^{-1})$and since $h(x^{-1})g(x^{-1}) = (x^{-1})^n - 1$

we have that $x^kh(x^{-1})x^{n-k}g(x^{-1}) = x^n(x^{-n} - 1) = 1 - x^n$

and therefore $\overline{h}(x)$ is indeed a factor of $x^n - 1$.

Encoding using a cyclic code can be done by a multiplication of two polynomials - a message polynomial and the generating polynomial for the cyclic code.

Let C be an $[n,k]$-code over an field F with the generator polynomial

$$g(x) = g_0 + g_1x + \ldots + g_{r-1}x^{r-1} \text{ of degree } r = n - k.$$

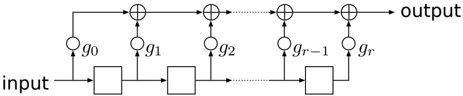
If a message vector m is represented by a polynomial $m(x)$ of degree k and m is encoded by

$$m \Rightarrow c = mG,$$

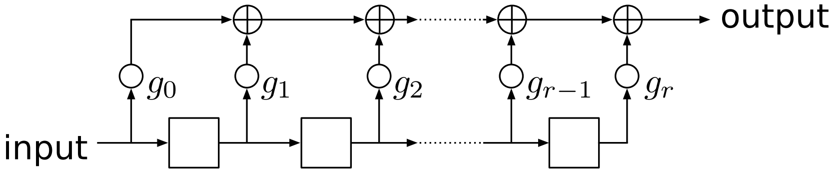
then the following relation between $m(x)$ and $c(x)$ holds

$$c(x) = m(x)g(x).$$

Such an encoding can be realized by the shift register shown in Figure below, where input is the k-bit message to be encoded followed by $n-k$ 0' and the output will be the encoded message.



Shift-register encodings of cyclic codes. Small circles represent multiplication by the corresponding constant, $\bigoplus$ nodes represent modular addition, squares are shift elements



Shift-register encodings of cyclic codes. Small circles represent multiplication by the corresponding constant, $\bigoplus$ nodes represent modular addition, squares are delay elements

Definition (Again!) Let r be a positive integer and let H be an $r \times (2^r - 1)$ matrix whose columns are all distinct non-zero vectors of $V(r, 2)$. Then the code having H as its parity-check matrix is called binary **Hamming code** denoted by *Ham* $(r, 2)$.

It can be shown that:

HAMMING CODES as CYCLIC CODES I

Definition (Again!) Let r be a positive integer and let H be an $r \times (2^r - 1)$ matrix whose columns are all distinct non-zero vectors of $V(r, 2)$. Then the code having H as its parity-check matrix is called binary **Hamming code** denoted by *Ham* $(r, 2)$.

It can be shown that:

Theorem The binary Hamming code *Ham* $(r, 2)$ is equivalent to a cyclic code.

HAMMING CODES as CYCLIC CODES I

Definition (Again!) Let r be a positive integer and let H be an $r \times (2^r - 1)$ matrix whose columns are all distinct non-zero vectors of $V(r, 2)$. Then the code having H as its parity-check matrix is called binary **Hamming code** denoted by $Ham\ (r, 2)$.

It can be shown that:

Theorem The binary Hamming code $Ham\ (r, 2)$ is equivalent to a cyclic code.

Definition If $p(x)$ is an irreducible polynomial of degree r such that x is a primitive element of the field $F[x]/p(x)$, then $p(x)$ is called a primitive polynomial.

Definition (Again!) Let r be a positive integer and let H be an $r \times (2^r - 1)$ matrix whose columns are all distinct non-zero vectors of $V(r, 2)$. Then the code having H as its parity-check matrix is called binary **Hamming code** denoted by $Ham\ (r, 2)$.

It can be shown that:

Theorem The binary Hamming code $Ham\ (r, 2)$ is equivalent to a cyclic code.

Definition If $p(x)$ is an irreducible polynomial of degree r such that x is a primitive element of the field $F[x]/p(x)$, then $p(x)$ is called a primitive polynomial.

Theorem If $p(x)$ is a primitive polynomial over $GF(2)$ of degree r, then the cyclic code $\langle p(x) \rangle$ is the code $Ham\ (r, 2)$.

HAMMING CODES as CYCLIC CODES II

Example Polynomial $x^3 + x + 1$ is irreducible over $GF(2)$ and x is primitive element of the field $F_2[x]/(x^3 + x + 1)$.

$$F_2[x]/(x^3 + x + 1) =$$

$$\{0, 1, x, x^2, x^3 = x + 1, x^4 = x^2 + x, x^5 = x^2 + x + 1, x^6 = x^2 + 1\}$$

The parity-check matrix for a cyclic version of $Ham\ (3, 2)$

$$H = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

PROOF of THEOREM

The binary Hamming code *Ham* $(r, 2)$ is equivalent to a cyclic code.
It is known from algebra that if $p(x)$ is an irreducible polynomial of degree r, then the ring $F_2[x]/p(x)$ is a field of order 2^r.
In addition, every finite field has a primitive element. Therefore, there exists an element α of $F_2[x]/p(x)$ such that

$$F_2[x]/p(x) = \{0, 1, \alpha, \alpha^2, \ldots, \alpha^{2r-2}\}.$$

Let us identify an element $a_0 + a_1 + \ldots a_{r-1}x^{r-1}$ of $F_2[x]/p(x)$ with the column vector

$$(a_0, a_1, \ldots, a_{r-1})^\top$$

and consider the binary $r \times (2^r - 1)$ matrix

$$H = [1\ \alpha\ \alpha^2 \ldots \alpha^{2^r-2}].$$

Let now C be the binary linear code having H as a parity check matrix.
Since the columns of H are all distinct non-zero vectors of $V(r, 2)$, $C = Ham\ (r, 2)$.
Putting $n = 2^r - 1$ we get

$$C = \{f_0 f_1 \ldots f_{n-1} \in V(n, 2) | f_0 + f_1\alpha + \ldots + f_{n-1}\alpha^{n-1} = 0\} \quad (1)$$

$$= \{f(x) \in R_n | f(\alpha) = 0 \text{ in } F_2[x]/p(x)\} \quad (2)$$

If $f(x) \in C$ and $r(x) \in R_n$, then $r(x)f(x) \in C$ because

$$r(\alpha)f(\alpha) = r(\alpha) \bullet 0 = 0$$

and therefore, by one of the previous theorems, this version of *Ham* $(r, 2)$ is cyclic.

To the most important cyclic codes for applications belong BCH codes and Reed-Solomon codes.

Definition A polynomial p is said to be minimal for a complex number x in Z_q if $p(x) = 0$ and p is irreducible over Z_q.

[1] BHC stands for Bose and Ray-Chaudhuri and Hocquenghem who discovered these codes.

To the most important cyclic codes for applications belong BCH codes and Reed-Solomon codes.

Definition A polynomial p is said to be minimal for a complex number x in Z_q if $p(x) = 0$ and p is irreducible over Z_q.

Definition A cyclic code of codewords of length n over Z_q, $q = p^r$, p is a prime, is called BCH code[1] of distance d if its generator $g(x)$ is the least common multiple of the minimal polynomials for

$$\omega^l, \omega^{l+1}, \ldots, \omega^{l+d-2}$$

for some l, where

ω is the primitive n-th root of unity.

If $n = q^m - 1$ for some m, then the BCH code is called primitive.

[1] BHC stands for Bose and Ray-Chaudhuri and Hocquenghem who discovered these codes.

To the most important cyclic codes for applications belong BCH codes and Reed-Solomon codes.

Definition A polynomial p is said to be minimal for a complex number x in Z_q if $p(x) = 0$ and p is irreducible over Z_q.

Definition A cyclic code of codewords of length n over Z_q, $q = p^r$, p is a prime, is called BCH code[1] of distance d if its generator $g(x)$ is the least common multiple of the minimal polynomials for

$$\omega^l, \omega^{l+1}, \dots, \omega^{l+d-2}$$

for some l, where

$$\omega \text{ is the primitive } n\text{-th root of unity.}$$

If $n = q^m - 1$ for some m, then the BCH code is called primitive.

Definition A Reed-Solomon code is a primitive BCH code with $n = q - 1$.

Properties:

- Reed-Solomon codes are self-dual.

[1] BHC stands for Bose and Ray-Chaudhuri and Hocquenghem who discovered these codes.

CHANNEL (STREAMS) CODING I.

The task of channel coding is to encode streams of data in such a way that if they are sent over a noisy channel errors can be detected and/or corrected by the receiver.

CHANNEL (STREAMS) CODING I.

The task of channel coding is to encode streams of data in such a way that if they are sent over a noisy channel errors can be detected and/or corrected by the receiver.

In case no receiver-to-sender communication is allowed we speak about **forward error correction**.

CHANNEL (STREAMS) CODING I.

The task of channel coding is to encode streams of data in such a way that if they are sent over a noisy channel errors can be detected and/or corrected by the receiver.

In case no receiver-to-sender communication is allowed we speak about **forward error correction**.

An important parameter of a channel code is **code rate**

$$r = \frac{k}{n}$$

in case k bits are encoded by n bits.

The code rate expressed the amount of redundancy in the code - the lower is the rate, the more redundant is the code.

CHANNEL (STREAM) CODING II

Design of a channel code is always a tradeoff between **energy efficiency** and **bandwidth efficiency**.

Codes with lower code rate can usually correct more errors. Consequently, the communication system can operate

- with a lower transmit power;
- transmit over longer distances;
- tolerate more interference;
- use smaller antennas;
- transmit at a higher data rate.

These properties make codes with lower code rate energy efficient.

On the other hand such codes require larger bandwidth and decoding is usually of higher complexity.

The selection of the code rate involves a tradeoff between energy efficiency and bandwidth efficiency.

Central problem of channel encoding: encoding is usually easy, but decoding is usually hard.

CONVOLUTION CODES

Our first example of channel codes are convolution codes.

Convolution codes have simple encoding and decoding, are quite a simple generalization of linear codes and have encodings as cyclic codes.

Our first example of channel codes are convolution codes.

Convolution codes have simple encoding and decoding, are quite a simple generalization of linear codes and have encodings as cyclic codes.

An (n, k) convolution code (CC) is defined by an $k \times n$ generator matrix, entries of which are polynomials over F_2.

For example,

$$G_1 = [x^2 + 1, x^2 + x + 1]$$

is the generator matrix for a $(2, 1)$ convolution code CC_1 and

$$G_2 = \begin{pmatrix} 1 + x & 0 & x + 1 \\ 0 & 1 & x \end{pmatrix}$$

is the generator matrix for a $(3, 2)$ convolution code CC_2

ENCODING of FINITE POLYNOMIALS

An (n,k) convolution code with a k x n generator matrix G can be used to encode a k-tuple of plain-polynomials (polynomial input information)

$$I = (I_0(x), I_1(x), \dots, I_{k-1}(x))$$

to get an n-tuple of crypto-polynomials

$$C = (C_0(x), C_1(x), \dots, C_{n-1}(x))$$

As follows

$$C = I \cdot G$$

EXAMPLES

EXAMPLE 1

$$
\begin{aligned}
(x^3+x+1)\cdot G_1 &= (x^3+x+1)\cdot(x^2+1, x^2+x+1)\\
&= (x^5+x^2+x+1, x^5+x^4+1)
\end{aligned}
$$

EXAMPLE 2

$$
(x^2+x, x^3+1)\cdot G_2 = (x^2+x, x^3+1)\cdot\begin{pmatrix}1+x & 0 & x+1\\ 0 & 1 & x\end{pmatrix}
$$

The way infinite streams are encoded using convolution codes will be Illustrated on the code CC_1.

An input stream $I = (I_0, I_1, I_2, \dots)$ is mapped into the output stream $C = (C_{00}, C_{10}, C_{01}, C_{11} \dots)$ defined by

$$C_0(x) = C_{00} + C_{01}x + \dots = (x^2 + 1)I(x)$$

and

$$C_1(x) = C_{10} + C_{11}x + \dots = (x^2 + x + 1)I(x).$$

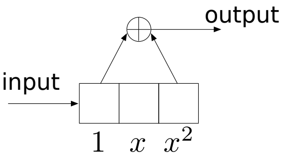
The first multiplication can be done by the first shift register from the next figure; second multiplication can be performed by the second shift register on the next slide and it holds

$$C_{0i} = I_i + I_{i+2}, \quad C_{1i} = I_i + I_{i-1} + I_{i-2}.$$

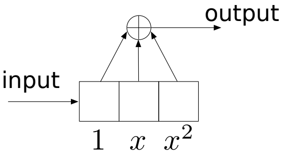
That is the output streams C_0 and C_1 are obtained by convolving the input stream with polynomials of G_1.

ENCODING

The first shift register

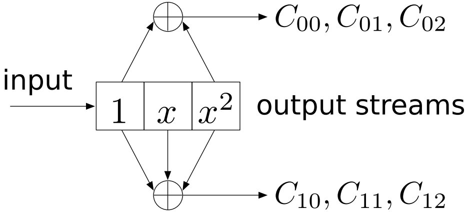


will multiply the input stream by $x^2 + 1$ and the second shift register



will multiply the input stream by $x^2 + x + 1$.

The following shift-register will therefore be an encoder for the code CC_1



For decoding of the convolution codes so called

Viterbi algorithm

Is used.

SHANNON CHANNEL CAPACITY

For every combination of bandwidth (W), channel type , signal power (S) and received noise power (N), there is a theoretical upper bound, called **channel capacity** or **Shannon capacity**, on the data transmission rate R for which error-free data transmission is possible.

SHANNON CHANNEL CAPACITY

For every combination of bandwidth (W), channel type , signal power (S) and received noise power (N), there is a theoretical upper bound, called **channel capacity** or **Shannon capacity**, on the data transmission rate R for which error-free data transmission is possible.

For so-called Additive White Gaussian Noise (AWGN) channels, that well capture deep space channels, this limit is (so-called Shannon-Hartley theorem):

$$R < W \log\left(1 + \frac{S}{N}\right) \quad \{\text{bits per second}\}$$

Shannon capacity sets a limit to the energy efficiency of the code.

SHANNON CHANNEL CAPACITY

For every combination of bandwidth (W), channel type , signal power (S) and received noise power (N), there is a theoretical upper bound, called **channel capacity** or **Shannon capacity**, on the data transmission rate R for which error-free data transmission is possible.

For so-called Additive White Gaussian Noise (AWGN) channels, that well capture deep space channels, this limit is (so-called Shannon-Hartley theorem):

$$R < W \log\left(1 + \frac{S}{N}\right) \quad \{\text{bits per second}\}$$

Shannon capacity sets a limit to the energy efficiency of the code.

Till 1993 channel code designers were unable to develop codes with performance close to Shannon capacity limit, that is Shannon capacity approaching codes, and practical codes required about twice as much energy as theoretical minimum predicted.

Therefore there was a big need for better codes with performance (arbitrarily) close to Shannon capacity limits.

Concatenated codes and Turbo codes have such a Shannon capacity approaching property.

Let $C_{in} : A^k \rightarrow A^n$ be an $[n, k, d]$ code over alphabet A.

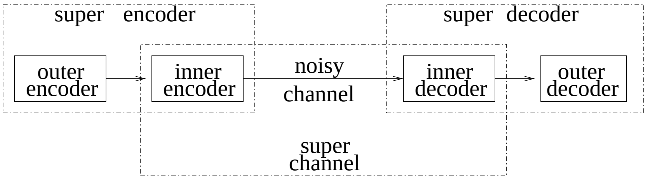
Let $C_{out} : B^K \rightarrow B^N$ be an $[N, K, D]$ code over alphabet B with $|B| = |A|^k$ symbols.

Concatenation of C_{out} (as outer code) with C_{in} (as inner code), denoted $C_{out} \circ C_{in}$ is the $[nN, kK, dD]$ code

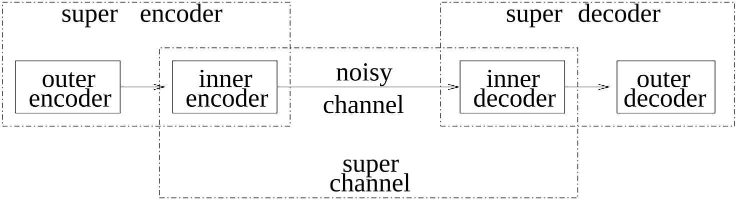
$$C_{out} \circ C_{in} : A^{kK} \rightarrow A^{nN}$$

that maps an input message $m = (m_1, m_2, \ldots, m_K)$ to a codeword $(C_{in}(m_1^{'}), C_{in}(m_2^{'}), \ldots, C_{in}(m_N^{'}))$, where

$$(m_1^{'}, m_2^{'}, \ldots, m_N^{'}) = C_{out}(m_1, m_2, \ldots, m_K)$$



CONCATENATED CODES

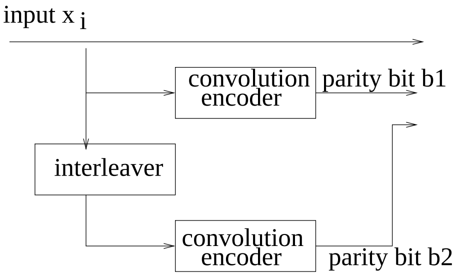


Of the key importance is the fact that if C_{in} is decoded using the *maximum-likelihood principle* (thus showing an exponentially decreasing error probability with increasing length) and C_{out} is a code with length $N = 2^n r$ that can be decoded in polynomial time in N, then the concatenated code can be decoded in polynomial time with respect to $n2^{nr}$ and has exponentially decreasing error probability even if C_{in} has exponential decoding complexity.

APPLICATIONS

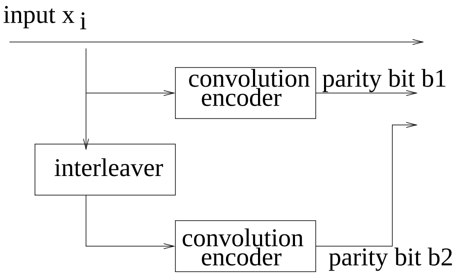
- Concatenated codes started to be used for deep space communication starting with Voyager program in 1977 and stayed so until the invention of Turbo codes and LDPC codes.
- Concatenated codes are used also on Compact Disc.
- The best concatenated codes for many applications were based on outer Reed-Solomon codes and inner Viterbi-decoded short constant length convolution codes.

Turbo codes were introduced by Berrou, Glavieux and Thitimajshima in 1993.
A Turbo code is formed from the parallel composition of two (convolution) codes separated by an interleaver (that permutes blocks of data in a fixed (pseudo)-random way).
A Turbo encoder is formed from the parallel composition of two (convolution) encoders separated by an interleaver.

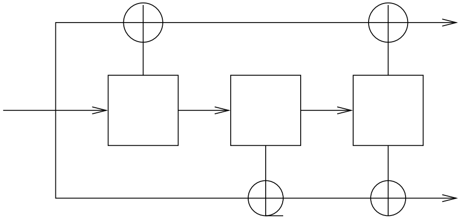


EXAMPLE of TURBO and CONVOLUTION ENCODERS

A Turbo encoder



and a convolution encoder



DECODING and PERFORMANCE of TURBO CODES

- A soft-in-soft-out decoding is used - the decoder gets from the analog/digital demodulator a soft value of each bit - probability that it is 1 and produces only a soft-value for each bit.
- The overall decoder uses decoders for outputs of two encoders that also provide only soft values for bits and by exchanging information produced by two decoders and from the original input bit, the main decoder tries to increase, by an iterative process, likelihood for values of decoded bits and to produce finally hard outcome - a bit 1 or 0.
- Turbo codes performance can be very close to theoretical Shannon limit.
- This was, for example the case for UMTS (the third Generation Universal Mobile Telecommunication System) Turbo code having a less than 1.2-fold overhead. in this case the interleaver worked with block of 40-5114 bits.
- Turbo codes were incorporated into standards used by NASA for deep space communications, digital video broadcasting and both third generation cellular standards.
- Literature: M.C. Valenti and J.Sun: Turbo codes - tutorial, Handbook of RF and Wireless Technologies, 2004 - reachable by Google.

REACHING SHANNON LIMIT

- Though Shannon developed his capacity bound already in 1940, till recently code designers were unable to come with codes with performance close to theoretical limit.
- In 1990 the gap between theoretical bound and practical implementations was still at best about 3dB A decibel is a relative measure. If E is the actual energy and E_{ref} is the theoretical lower bound, then the relative energy increase in decibels is

$$10 \log_{10} \frac{E}{E_{ref}}$$

  Since $\log_{10} 2 = 0.3$ a two-fold relative energy increase equals $3dB$.
- For code rate $\frac{1}{2}$ the relative increase in energy consumption is about 4.8 dB for convolution codes and 0.98 for Turbo codes.

WHY ARE TURBO CODES SO GOOD?

- Turbo codes are linear codes.
- A "good" linear code is one that has mostly high-weight codewords.
- High-weight codewords are desirable because they are more distinct and the decoder can more easily distinguish among them.
- A big advantage of Turbo encoders is that they reduce the number of low-weight codewords because their output is the sum of the weights of the input and two parity output bits.

Part IV

Secret-key cryptosystems

CHAPTER 4: CLASSICAL (SECRET-KEY) CRYPTOSYSTEMS

- In this chapter we deal with some of the very old or quite old classical (secret-key or symmetric) cryptosystems that were primarily used in the pre-computer era.

CHAPTER 4: CLASSICAL (SECRET-KEY) CRYPTOSYSTEMS

- In this chapter we deal with some of the very old or quite old classical (secret-key or symmetric) cryptosystems that were primarily used in the pre-computer era.
- These cryptosystems are too weak nowadays, too easy to break, especially with computers.

CHAPTER 4: CLASSICAL (SECRET-KEY) CRYPTOSYSTEMS

- In this chapter we deal with some of the very old or quite old classical (secret-key or symmetric) cryptosystems that were primarily used in the pre-computer era.
- These cryptosystems are too weak nowadays, too easy to break, especially with computers.
- However, these simple cryptosystems give a good illustration of several of the important ideas of the cryptography and cryptanalysis.
- Moreover, most of them can be very useful in combination with more modern cryptosystem - to add a new level of security.

Cryptology (= cryptography + cryptanalysis)
has more than two thousand years of history.

Cryptology (= cryptography + cryptanalysis)
has more than two thousand years of history.

Basic historical observation

- People have always had fascination with keeping information away from others.
- Some people – rulers, diplomats, military people, businessmen – have always had needs to keep some information away from others.

Cryptology (= cryptography + cryptanalysis)
has more than two thousand years of history.

Basic historical observation

- People have always had fascination with keeping information away from others.
- Some people – rulers, diplomats, military people, businessmen – have always had needs to keep some information away from others.

Importance of cryptography nowadays

- Applications: cryptography is the key tool to make modern information transmission secure, and to create secure information society.
- Foundations: cryptography gave rise to several new key concepts of the foundation of informatics: one-way functions, computationally perfect pseudorandom generators, zero-knowledge proofs, holographic proofs, program self-testing and self-correcting, . . .

Sound approaches to cryptography

- Shannon's approach based on information theory (enemy has not enough information to break a cryptosystem).
- Current approach based on complexity theory (enemy has not enough computation power to break a cryptosystem).
- Very recent approach based on the laws and limitations of quantum physics (enemy would need to break laws of nature to break a cryptosystem).

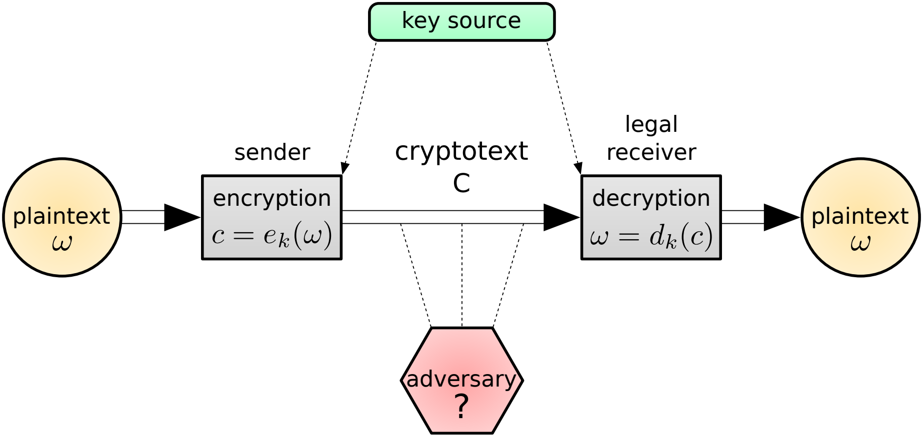
Sound approaches to cryptography

- Shannon's approach based on information theory (enemy has not enough information to break a cryptosystem).
- Current approach based on complexity theory (enemy has not enough computation power to break a cryptosystem).
- Very recent approach based on the laws and limitations of quantum physics (enemy would need to break laws of nature to break a cryptosystem).

Paradoxes of modern cryptography

- Positive results of modern cryptography are based on negative results of complexity theory.
- Computers, that were designed originally for decryption, seem to be now more useful for encryption.

The cryptography deals with problem of sending a message (plaintext, cleartext), through an insecure channel, that may be tapped by an adversary (eavesdropper, cryptanalyst), to a legal receiver.



COMPONENTS of CRYPTOSYSTEMS:

Plaintext-space: P – a set of plaintexts over an alphabet $\sum$

Cryptotext-space: C – a set of cryptotexts (ciphertexts) over alphabet Δ

Key-space: K – a set of keys

COMPONENTS of CRYPTOSYSTEMS:

Plaintext-space: P – a set of plaintexts over an alphabet $\sum$

Cryptotext-space: C – a set of cryptotexts (ciphertexts) over alphabet Δ

Key-space: K – a set of keys

Each key k determines an encryption algorithm e_k and an decryption algorithm d_k such that, for any plaintext $w, e_k(w)$ is the corresponding cryptotext and

$$w \in d_k(e_k(w)) \quad \text{or} \quad w = d_k(e_k(w)).$$

Note: As encryption algorithms we can use also randomized algorithms.

CAESAR can be used to encrypt words in any alphabet.

In order to encrypt words in English alphabet we use:

CAESAR can be used to encrypt words in any alphabet.

In order to encrypt words in English alphabet we use:

Key-space: $\{0, 1, \ldots, 25\}$

CAESAR can be used to encrypt words in any alphabet.

In order to encrypt words in English alphabet we use:

Key-space: $\{0, 1, \dots, 25\}$

An encryption algorithm e_k substitutes any letter by the letter occurring k positions ahead (cyclically) in the alphabet.

100 – 42 B.C., CAESAR CRYPTOSYSTEM - SHIFT CIPHER I

CAESAR can be used to encrypt words in any alphabet.

In order to encrypt words in English alphabet we use:

Key-space: $\{0, 1, \ldots, 25\}$

An encryption algorithm e_k substitutes any letter by the letter occurring k positions ahead (cyclically) in the alphabet.

A decryption algorithm d_k substitutes any letter by the one occurring **k** positions backward (cyclically) in the alphabet.

Example
$e_2(\text{EXAMPLE}) = \text{GZCOSNG},$
$e_2(\text{EXAMPLE}) = \text{HADPTOH},$
$e_1(\text{HAL}) = \text{IBM},$
$e_3(\text{COLD}) = \text{FROG}$

ABCDEFGHIJKLMNOPQRSTUVWXYZ

Example $e_2(\text{EXAMPLE}) = \text{GZCOSNG}$,
$e_2(\text{EXAMPLE}) = \text{HADPTOH}$,
$e_1(\text{HAL}) = \text{IBM}$,
$e_3(\text{COLD}) = \text{FROG}$

ABCDEFGHIJKLMNOPQRSTUVWXYZ

Example Find the plaintext to the following cryptotext obtained by the encryption with CAESAR with $\mathbf{k} = ?$.

Cryptotext: VHFUHW GH GHXA, VHFUHW GH GLHX,
VHFUHW GH WURLV, VHFUHW GH WRXV.

100 – 42 B.C., CAESAR CRYPTOSYSTEM - SHIFT CIPHER II

Example $e_2(\text{EXAMPLE}) = \text{GZCOSNG}$,
$e_2(\text{EXAMPLE}) = \text{HADPTOH}$,
$e_1(\text{HAL}) = \text{IBM}$,
$e_3(\text{COLD}) = \text{FROG}$

ABCDEFGHIJKLMNOPQRSTUVWXYZ

Example Find the plaintext to the following cryptotext obtained by the encryption with CAESAR with $\mathbf{k} = ?$.

Cryptotext: VHFUHW GH GHXA, VHFUHW GH GLHX,
VHFUHW GH WURLV, VHFUHW GH WRXV.

Numerical version of CAESAR is defined on the set $\{0, 1, 2, \ldots, 25\}$ by the encryption algorithm:

$$e_k(i) = (i + k)(mod\ 26)$$

POLYBIOUS CRYPTOSYSTEM

for encryption of words of the English alphabet without J.

Key-space: Polybious checkerboards 5×5 with 25 English letters and with rows + columns labeled by symbols.

Encryption algorithm: Each symbol is substituted by the pair of symbols denoting the row and the column of the checkerboard in which the symbol is placed.

Example:

	F	G	H	I	J
A	A	B	C	D	E
B	F	G	H	I	K
C	L	M	N	O	P
D	Q	R	S	T	U
E	V	W	X	Y	Z

KONIEC $\rightarrow$

Decryption algorithm: ???

KERCKHOFF's PRINCIPLE

The philosophy of modern cryptanalysis is embodied in the following principle formulated in 1883 by Jean Guillaume Hubert Victor Francois Alexandre Auguste Kerckhoffs von Nieuwenhof (1835 - 1903).

KERCHKHOFF's PRINCIPLE

The philosophy of modern cryptanalysis is embodied in the following principle formulated in 1883 by Jean Guillaume Hubert Victor Francois Alexandre Auguste Kerckhoffs von Nieuwenhof (1835 - 1903).

The security of a cryptosystem must not depend on keeping secret the encryption algorithm. The security should depend only on *keeping secret the key.*

(Sir Francis R. Bacon (1561 - 1626))

1. Given e_k and a plaintext w, it should be easy to compute $c = e_k(w)$.

REQUIREMENTS for GOOD CRYPTOSYSTEMS

(Sir Francis R. Bacon (1561 - 1626))

1. Given e_k and a plaintext w, it should be easy to compute $c = e_k(w)$.
2. Given d_k and a cryptotext c, it should be easy to compute $w = d_k(c)$.

REQUIREMENTS for GOOD CRYPTOSYSTEMS

(Sir Francis R. Bacon (1561 - 1626))

1. Given e_k and a plaintext w, it should be easy to compute $c = e_k(w)$.
2. Given d_k and a cryptotext c, it should be easy to compute $w = d_k(c)$.
3. A cryptotext $e_k(w)$ should not be much longer than the plaintext w.

REQUIREMENTS for GOOD CRYPTOSYSTEMS

(Sir Francis R. Bacon (1561 - 1626))

1. Given e_k and a plaintext w, it should be easy to compute $c = e_k(w)$.
2. Given d_k and a cryptotext c, it should be easy to compute $w = d_k(c)$.
3. A cryptotext $e_k(w)$ should not be much longer than the plaintext w.
4. It should be unfeasible to determine w from $e_k(w)$ without knowing d_k.

(Sir Francis R. Bacon (1561 - 1626))

1. Given e_k and a plaintext w, it should be easy to compute $c = e_k(w)$.
2. Given d_k and a cryptotext c, it should be easy to compute $w = d_k(c)$.
3. A cryptotext $e_k(w)$ should not be much longer than the plaintext w.
4. It should be unfeasible to determine w from $e_k(w)$ without knowing d_k.
5. The so called avalanche effect should hold: A small change in the plaintext, or in the key, should lead to a big change in the cryptotext (i.e. a change of one bit of the plaintext should result in a change of all bits of the cryptotext, each with the probability close to 0.5).

(Sir Francis R. Bacon (1561 - 1626))

1. Given e_k and a plaintext w, it should be easy to compute $c = e_k(w)$.
2. Given d_k and a cryptotext c, it should be easy to compute $w = d_k(c)$.
3. A cryptotext $e_k(w)$ should not be much longer than the plaintext w.
4. It should be unfeasible to determine w from $e_k(w)$ without knowing d_k.
5. The so called avalanche effect should hold: A small change in the plaintext, or in the key, should lead to a big change in the cryptotext (i.e. a change of one bit of the plaintext should result in a change of all bits of the cryptotext, each with the probability close to 0.5).
6. The cryptosystem should not be closed under composition, i.e. not for every two keys k_1, k_2 there is a key k such that

$$e_k(w) = e_{k_1}(e_{k_2}(w)).$$

(Sir Francis R. Bacon (1561 - 1626))

1. Given e_k and a plaintext w, it should be easy to compute $c = e_k(w)$.
2. Given d_k and a cryptotext c, it should be easy to compute $w = d_k(c)$.
3. A cryptotext $e_k(w)$ should not be much longer than the plaintext w.
4. It should be unfeasible to determine w from $e_k(w)$ without knowing d_k.
5. The so called avalanche effect should hold: A small change in the plaintext, or in the key, should lead to a big change in the cryptotext (i.e. a change of one bit of the plaintext should result in a change of all bits of the cryptotext, each with the probability close to 0.5).
6. The cryptosystem should not be closed under composition, i.e. not for every two keys k_1, k_2 there is a key k such that
$$e_k(w) = e_{k_1}(e_{k_2}(w)).$$
7. The set of keys should be very large.

The aim of cryptanalysis is to get as much information about the plaintext or the key as possible.

Main types of cryptanalytic attacks

1. Cryptotexts-only attack. The cryptanalysts get cryptotexts $c_1 = e_k(w_1), \dots, c_n = e_k(w_n)$ and try to infer the key k or as many of the plaintexts $w_1, \dots, w_n$ as possible.

The aim of cryptanalysis is to get as much information about the plaintext or the key as possible.

Main types of cryptanalytic attacks

1. Cryptotexts-only attack. The cryptanalysts get cryptotexts $c_1 = e_k(w_1), \ldots, c_n = e_k(w_n)$ and try to infer the key k or as many of the plaintexts $w_1, \ldots, w_n$ as possible.
2. Known-plaintexts attack **(given are some pairs [plaintext, cryptotext])** The cryptanalysts know some pairs $w_i, e_k(w_i), 1 \leq i \leq n$, and try to infer k, or at least w_{n+1} for a new cryptotext $e_k(w_{n+1})$.

The aim of cryptanalysis is to get as much information about the plaintext or the key as possible.

Main types of cryptanalytic attacks

1. Cryptotexts-only attack. The cryptanalysts get cryptotexts $c_1 = e_k(w_1), \ldots, c_n = e_k(w_n)$ and try to infer the key k or as many of the plaintexts $w_1, \ldots, w_n$ as possible.
2. Known-plaintexts attack **(given are some pairs [plaintext, cryptotext])** The cryptanalysts know some pairs $w_i, e_k(w_i), 1 \leq i \leq n$, and try to infer k, or at least w_{n+1} for a new cryptotext $e_k(w_{n+1})$.
3. Chosen-plaintexts attack (given are cryptotext for some chosen plaintexts) The cryptanalysts choose plaintexts $w_1, \ldots, w_n$ to get cryptotexts $e_k(w_1), \ldots, e_k(w_n)$, and try to infer k or at least w_{n+1} for a new cryptotext $c_{n+1} = e_k(w_{n+1})$. (For example, if they get temporary access to the encryption machinery.)

4. Known-encryption-algorithm attack
The encryption algorithm e_k is given and the cryptanalysts try to get the decryption algorithm d_k.

4. Known-encryption-algorithm attack
   The encryption algorithm e_k is given and the cryptanalysts try to get the decryption algorithm d_k.
5. Chosen-cryptotext attack (given are plaintexts for some chosen cryptotexts)
   The cryptanalysts know some pairs
   $$[c_i, d_k(c_i)], \quad 1 \leq i \leq n,$$
   where the cryptotexts c_i have been chosen by the cryptanalysts. The aim is to determine the key. (For example, if cryptanalysts get a temporary access to decryption machinery.)

Let us assume that a clever Alice sends an encrypted message to Bob.
What can a bad enemy, called usually Eve (eavesdropper), do?

- Eve can read (and try to decrypt) the message.
- Eve can try to get the key that was used and then decrypt all messages encrypted with the same key.
- Eve can change the message sent by Alice into another message, in such a way that Bob will have the feeling, after he gets the changed message, that it was a message from Alice.
- Eve can pretend to be Alice and communicate with Bob, in such a way that Bob thinks he is communicating with Alice.

An eavesdropper can therefore be passive - Eve or active - Mallot.

BASIC GOALS of BROADLY UNDERSTOOD CRYPTOGRAPHY

Confidentiality: Eve should not be able to decrypt the message Alice sends to Bob.

Data integrity: Bob wants to be sure that Alice's message has not been altered by Eve.

Authentication: Bob wants to be sure that only Alice could have sent the message he has received.

Non-repudiation: Alice should not be able to claim that she did not send messages that she has sent.

Anonymity: Alice does not want Bob to find out who sent the message

HILL CRYPTOSYSTEM I

The cryptosystem presented in this slide was probably never used. In spite of that this cryptosystem played an important role in the history of modern cryptography.

We describe Hill cryptosystem for a fixed n and the English alphabet.

Key-space: The set of all matrices M of degree n with elements from the set $\{0, 1, \dots, 25\}$ such that $M^{-1} mod$ 26 exist.

Plaintext + cryptotext space: English words of length n.

Encoding: For a word w let c_w be the column vector of length n of the integer codes of symbols of w. $(A \rightarrow 0, B \rightarrow 1, C \rightarrow 2, \dots)$

Encryption: $c_c = Mc_w \bmod 26$

Decryption: $c_w = M^{-1}c_c \bmod 26$

HILL CRYPTOSYSTEM - EXAMPLE

Example A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

$$M = \begin{bmatrix} 4 & 7 \\ 1 & 1 \end{bmatrix} M^{-1} = \begin{bmatrix} 17 & 11 \\ 9 & 16 \end{bmatrix}$$

Plaintext: $w =$ LONDON

$$C_{LO} = \begin{bmatrix} 11 \\ 14 \end{bmatrix}, C_{ND} = \begin{bmatrix} 13 \\ 3 \end{bmatrix}, C_{ON} = \begin{bmatrix} 14 \\ 13 \end{bmatrix}$$

$$MC_{LO} = \begin{bmatrix} 12 \\ 25 \end{bmatrix}, MC_{ND} = \begin{bmatrix} 21 \\ 16 \end{bmatrix}, MC_{ON} = \begin{bmatrix} 17 \\ 1 \end{bmatrix}$$

Cryptotext: MZVQRB

Theorem

$$\text{If } M = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \text{ then} M^{-1} = \frac{1}{\det\ M} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix}$$

Proof: Exercise

SECRET-KEY (SYMMETRIC) CRYPTOSYSTEMS

A cryptosystem is called secret-key cryptosystem if some secret piece of information – the key – has to be agreed first between any two parties that have, or want, to communicate through the cryptosystem. Example: CAESAR, HILL. Another name is symmetric cryptosystem (cryptography).

SECRET-KEY (SYMMETRIC) CRYPTOSYSTEMS

A cryptosystem is called secret-key cryptosystem if some secret piece of information – the key – has to be agreed first between any two parties that have, or want, to communicate through the cryptosystem. Example: CAESAR, HILL. Another name is symmetric cryptosystem (cryptography).

Two basic types of secret-key cryptosystems

- **substitution** based cryptosystems
- **transposition** based cryptosystems

SECRET-KEY (SYMMETRIC) CRYPTOSYSTEMS

A cryptosystem is called secret-key cryptosystem if some secret piece of information – the key – has to be agreed first between any two parties that have, or want, to communicate through the cryptosystem. Example: CAESAR, HILL. Another name is symmetric cryptosystem (cryptography).

Two basic types of secret-key cryptosystems

- **substitution** based cryptosystems
- **transposition** based cryptosystems

Two basic types of substitution cryptosystems

- **monoalphabetic cryptosystems** – they use a fixed substitution – CAESAR, POLYBIOUS
- **polyalphabetic cryptosystems** – substitution keeps changing during the encryption

SECRET-KEY (SYMMETRIC) CRYPTOSYSTEMS

A cryptosystem is called secret-key cryptosystem if some secret piece of information – the key – has to be agreed first between any two parties that have, or want, to communicate through the cryptosystem. Example: CAESAR, HILL. Another name is symmetric cryptosystem (cryptography).

Two basic types of secret-key cryptosystems

- **substitution** based cryptosystems
- **transposition** based cryptosystems

Two basic types of substitution cryptosystems

- **monoalphabetic cryptosystems** – they use a fixed substitution – CAESAR, POLYBIOUS
- **polyalphabetic cryptosystems** – substitution keeps changing during the encryption

A monoalphabetic cryptosystem with letter-by-letter substitution is uniquely specified by a permutation of letters, (number of permutations (keys) is 26!)

Example: An AFFINE cryptosystem is given by two integers

$$0 \leq a, b \leq 25, gcd(a, 26) = 1.$$

Encryption: $e_{a,b}(x) = (ax + b) \bmod 26$

Example

$a = 3, b = 5, e_{3,5}(x) = (3x + 5) \bmod 26,$
$e_{3,5}(3) = 14, e_{3,5}(15) = 24 - e_{3,5}(D) = O, e_{3,5}(P) = Y$

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Decryption: $d_{a,b}(y) = a^{-1}(y - b) \bmod 26$

The basic cryptanalytic attack against monoalphabetic substitution cryptosystems begins with a frequency count: the number of each letter in the cryptotext is counted. The distributions of letters in the cryptotext is then compared with some official distribution of letters in the plaintext language.

The letter with the highest frequency in the cryptotext is likely to be substitute for the letter with highest frequency in the plaintext language .... The likelihood grows with the length of cryptotext.

Frequency counts in English:

	%		%		%
E	12.31	L	4.03	B	1.62
T	9.59	D	3.65	G	1.61
A	8.05	C	3.20	V	0.93
O	7.94	U	3.10	K	0.52
N	7.19	P	2.29	Q	0.20
I	7.18	F	2.28	X	0.20
S	6.59	M	2.25	J	0.10
R	6.03	W	2.03	Z	0.09
H	5.14	Y	1.88		
	70.02		24.71		5.27

and for other languages:

English	%	German	%	Finnish	%	French	%	Italian	%	Spanish	%
E	12.31	E	18.46	A	12.06	E	15.87	E	11.79	E	13.15
T	9.59	N	11.42	I	10.59	A	9.42	A	11.74	A	12.69
A	8.05	I	8.02	T	9.76	I	8.41	I	11.28	O	9.49
O	7.94	R	7.14	N	8.64	S	7.90	O	9.83	S	7.60
N	7.19	S	7.04	E	8.11	T	7.29	N	6.88	N	6.95
I	7.18	A	5.38	S	7.83	N	7.15	L	6.51	R	6.25
S	6.59	T	5.22	L	5.86	R	6.46	R	6.37	I	6.25
R	6.03	U	5.01	O	5.54	U	6.24	T	5.62	L	5.94
H	5.14	D	4.94	K	5.20	L	5.34	S	4.98	D	5.58

The 20 most common digrams are (in decreasing order) TH, HE, IN, ER, AN, RE, ED, ON, ES, ST, EN, AT, TO, NT, HA, ND, OU, EA, NG, AS. The six most common trigrams: THE, ING, AND, HER, ERE, ENT.

CRYPTANALYSIS of AFFINE CRYPTOSYSTEM - EXAMPLE

Cryptanalysis of a cryptotext encrypted using the AFFINE cryptosystem with an encryption algorithm

$$e_{a,b}(x) = (ax + b) \bmod 26 = (xa + b) \bmod 26$$

where $0 \leq a, b \leq 25, gcd(a, 26) = 1$. (Number of keys: $12 \times 26 = 312$.)

Example: Assume that an English plaintext is divided into blocks of 5 letters and encrypted by an AFFINE cryptosystem (ignoring space and interpunctions) as follows:

How to find the plaintext?

```
B H J U H   N B U L S   V U L R U   S L Y X H
O N U U N   B W N U A   X U S N L   U Y J S S
W X R L K   G N B O N   U U N B W   S W X K X
H K X D H   U Z D L K   X B H J U   H B N U O
N U M H U   G S W H U   X M B X R   W X K X L
U X B H J   U H C X K   X A X K Z   S W K X X
L K O L J   K C X L C   M X O N U   U B V U L
R R W H S   H B H J U   H N B X M   B X R W X
K X N O Z   L J B X X   H B N F U   B H J U H
L U S W X   G L L K Z   L J P H U   U L S Y X
B J K X S   W H S S W   X K X N B   H B H J U
H Y X W N   U G S W X   G L L K
```

Frequency analysis of plainext and frequency table for English:

X - 32	J - 11	D - 2
U - 30	O - 6	V - 2
H - 23	R - 6	F - 1
B - 19	G - 5	P - 1
L - 19	M - 4	E - 0
N - 16	Y - 4	I - 0
K - 15	Z - 4	Q - 0
S - 15	C - 3	T - 0
W - 14	A - 2	

	%		%		%
E	12.31	L	4.03	B	1.62
T	9.59	D	3.65	G	1.61
A	8.05	C	3.20	V	0.93
O	7.94	U	3.10	K	0.52
N	7.19	P	2.29	Q	0.20
I	7.18	F	2.28	X	0.20
S	6.59	M	2.25	J	0.10
R	6.03	W	2.03	Z	0.09
H	5.14	Y	1.88		
	70.02		24.71		5.27

First guess: $E = X, T = U$

Encodings: $4a + b = 23 \pmod{26}$

$xa + b = y$ $\quad 19a + b = 20 \pmod{26}$

Solutions: $a = 5, b = 3 \rightarrow a^{-1} =$

CRYPTANALYSIS - CONTINUATION I

Frequency analysis of plainext and frequency table for English:

X - 32	J - 11	D - 2
U - 30	O - 6	V - 2
H - 23	R - 6	F - 1
B - 19	G - 5	P - 1
L - 19	M - 4	E - 0
N - 16	Y - 4	I - 0
K - 15	Z - 4	Q - 0
S - 15	C - 3	T - 0
W - 14	A - 2	

	%		%		%
E	12.31	L	4.03	B	1.62
T	9.59	D	3.65	G	1.61
A	8.05	C	3.20	V	0.93
O	7.94	U	3.10	K	0.52
N	7.19	P	2.29	Q	0.20
I	7.18	F	2.28	X	0.20
S	6.59	M	2.25	J	0.10
R	6.03	W	2.03	Z	0.09
H	5.14	Y	1.88		
	70.02		24.71		5.27

First guess: $E = X, T = U$

Encodings: $xa + b = y$

$$4a + b = 23 \pmod{26}$$
$$19a + b = 20 \pmod{26}$$

Solutions: $a = 5, b = 3 \rightarrow a^{-1} =$

Translation table

crypto	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
plain	P	K	F	A	V	Q	L	G	B	W	R	M	H	C	X	S	N	I	D	Y	T	O	J	E	Z	U

```
B H J U H   N B U L S   V U L R U   S L Y X H
O N U U N   B W N U A   X U S N L   U Y J S S
W X R L K   G N B O N   U U N B W   S W X K X
H K X D H   U Z D L K   X B H J U   H B N U O
N U M H U   G S W H U   X M B X R   W X K X L
U X B H J   U H C X K   X A X K Z   S W K X X
L K O L J   K C X L C   M X O N U   U B V U L
R R W H S   H B H J U   H N B X M   B X R W X
K X N O Z   L J B X X   H B N F U   B H J U H
L U S W X   G L L K Z   L J P H U   U L S Y X
B J K X S   W H S S W   X K X N B   H B H J U
H Y X W N   U G S W X   G L L K
```

provides from the above cryptotext the plaintext that starts with KGWTG CKTMO OTMIT DMZEG, which does not make sense.

CRYPTANALYSIS - CONTINUATION II

Second guess: $E = X, A = H$

Equations

$$4a + b = 23 \pmod{26}$$
$$b = 7 \pmod{26}$$

Solutions: $a = 4$ or $a = 17$ and therefore $a = 17$

Second guess: $E = X, A = H$

Equations $$4a + b = 23 \pmod{26}$$
$$b = 7 \pmod{26}$$

Solutions: $a = 4$ or $a = 17$ and therefore $a = 17$

This gives the translation table

crypto	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
plain	V	S	P	M	J	G	D	A	X	U	R	O	L	I	F	C	Z	W	T	Q	N	K	H	E	B	Y

and the following plaintext from the above cryptotext

```
SAUNA ISNOT KNOWN TOBEA
FINNI SHINV ENTIO NBUTT
HEWOR DISFI NNISH THERE
AREMA NYMOR ESAUN ASINF
INLAN DTHAN ELSEW HEREO
NESAU NAPER EVERY THREE
ORFOU RPEOP LEFIN NSKNO
WWHAT ASAUN AISEL SEWHE
REIFY OUSEE ASIGN SAUNA
ONTHE DOORY OUCAN NOTBE
SURET HATTH EREIS ASAUN
ABEHI NDTHE DOOR
```

EXAMPLES of MONOALPHABETIC CRYPTOSYSTEMS

Symbols of the English alphabet will be replaced by squares with or without points and with or without surrounding lines using the following rule:

A:	B:	C:
D:	E:	F:
G:	H:	I:

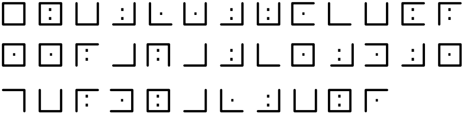
J·	K·	L·
M·	N·	O·
P·	Q·	R·

S	T	U
V	W	X
Y	Z	

For example the plaintext:

WE TALK ABOUT FINNISH SAUNA MANY TIMES LATER

results in the cryptotext:



EXAMPLES of MONOALPHABETIC CRYPTOSYSTEMS

Symbols of the English alphabet will be replaced by squares with or without points and with or without surrounding lines using the following rule:

A:	B:	C:
D:	E:	F:
G:	H:	I:

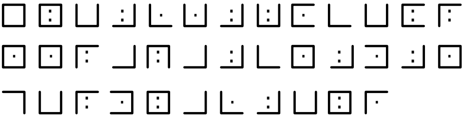
J·	K·	L·
M·	N·	O·
P·	Q·	R·

S	T	U
V	W	X
Y	Z	

For example the plaintext:

WE TALK ABOUT FINNISH SAUNA MANY TIMES LATER

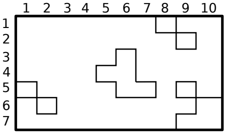
results in the cryptotext:



Garbage in between method: the message (plaintext or cryptotext) is supplemented by "garbage letters".

Richelieu cryptosystem used sheets of card board with holes.

I		L	O	V	E		Y	O	U
I		H	A	V	E		Y	O	U
D	E	E	P		U	N	D	E	R
M	Y		S	K	I	N		M	Y
L	O	V	E		L	A	S	T	S
F	O	R	E	V	E	R		I	N
H	Y	P	E	R	S	P	A	C	E



Playfair cryptosystem

Invented around 1854 by Ch. Wheatstone.

Key – a Playfair square is defined by a word w of length at most 25. In w repeated letters are then removed, remaining letters of alphabets (except j) are then added and resulting word is divided to form an 5 x 5 array (a Playfair square).

Playfair cryptosystem

Invented around 1854 by Ch. Wheatstone.

Key – a Playfair square is defined by a word w of length at most 25. In w repeated letters are then removed, remaining letters of alphabets (except j) are then added and resulting word is divided to form an 5 x 5 array (a Playfair square).

Encryption: of a pair of letters x, y

1. If x and y are in the same row (column), then they are replaced by the pair of symbols to the right (bellow) them.
2. If x and y are in different rows and columns they are replaced by symbols in the opposite corners of rectangle created by x and y. the rder is important.

Playfair cryptosystem

Invented around 1854 by Ch. Wheatstone.

Key – a Playfair square is defined by a word w of length at most 25. In w repeated letters are then removed, remaining letters of alphabets (except j) are then added and resulting word is divided to form an 5 x 5 array (a Playfair square).

Encryption: of a pair of letters x, y

1. If x and y are in the same row (column), then they are replaced by the pair of symbols to the right (bellow) them.
2. If x and y are in different rows and columns they are replaced by symbols in the opposite corners of rectangle created by x and y. the rder is important.

Example: PLAYFAIR is encrypted as LCMNNFCS
Playfair was used in World War I by British army.

Playfair square:

S	D	Z	I	U
H	A	F	N	G
B	M	V	Y	W
R	P	L	C	X
T	O	E	K	Q

VIGENERE and AUTOCLAVE cryptosystems

Several of the following polyalphabetic cryptosystems are modification of the CAESAR cryptosystem.

A 26×26 table is first designed with the first row containing a permutation of all symbols of alphabet and all columns represent CAESAR shifts starting with the symbol of the first row.

Secondly, for a plaintext w a key k is a word of the same length as w.

Encryption: the *i-th* letter of the plaintext - w_i is replaced by the letter in the w_i-row and k_i-column of the table.

VIGENERE and AUTOCLAVE cryptosystems

Several of the following polyalphabetic cryptosystems are modification of the CAESAR cryptosystem.

A 26×26 table is first designed with the first row containing a permutation of all symbols of alphabet and all columns represent CAESAR shifts starting with the symbol of the first row.

Secondly, for a plaintext w a key k is a word of the same length as w.

Encryption: the *i-th* letter of the plaintext - w_i is replaced by the letter in the w_i-row and k_i-column of the table.
VIGENERE cryptosystem: a short keyword p is chosen and

$$k = Prefix_{|w|}p^{oo}$$

VIGENERE is actually a cyclic version of the CAESAR cryptosystem.

VIGENERE and AUTOCLAVE cryptosystems

Several of the following polyalphabetic cryptosystems are modification of the CAESAR cryptosystem.

A 26×26 table is first designed with the first row containing a permutation of all symbols of alphabet and all columns represent CAESAR shifts starting with the symbol of the first row.

Secondly, for a plaintext w a key k is a word of the same length as w.

Encryption: the *i-th* letter of the plaintext - w_i is replaced by the letter in the w_i-row and k_i-column of the table.

VIGENERE cryptosystem: a short keyword p is chosen and

$$k = Prefix_{|w|} p^{oo}$$

VIGENERE is actually a cyclic version of the CAESAR cryptosystem.

AUTOCLAVE cryptosystem: $k = Prefix_{|w|} pw$

VIGENERE and AUTOCLAVE cryptosystems

Example:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Keyword: H A M B U R G
Plaintext: I N J E D E M M E N S C H E N G E S I C H T E S T E H T S E I N E G
Vigenere-key: H A M B U R G H A M B U R G H A M B U R G H A M B U R G H A M B U R
Autoclave-key: H A M B U R G I N J E D E M M E N S C H E N G E S I C H T E S T E H
Vigenere-cryp.: P N V F X V S T E Z T W Y K U G Q T C T N A E E U Y Y Z Z E U O Y X
Autoclave-cryp.: P N V F X V S U R W W F L Q Z K R K K J L G K W L M J A L I A G I N

1. Task 1 – to find the length of the key

Kasiski method (1852) - invented also by Charles Babbage (1853).

Basic observation If a subword of a plaintext is repeated at a distance that is a multiple of the length of the key, then the corresponding subwords of the cryptotext are the same.

1. Task 1 – to find the length of the key

Kasiski method (1852) - invented also by Charles Babbage (1853).

Basic observation If a subword of a plaintext is repeated at a distance that is a multiple of the length of the key, then the corresponding subwords of the cryptotext are the same.

Example, cryptotext:

CHRGQPWOEIRULYANDOSHCHRIZKEBUSNOFKYWROPDCHRKGAXBNRHROAKERBKSCHRIWK

Substring "CHR" occurs in positions 1, 21, 41, 66: expected keyword length is therefore 5.

1. Task 1 – to find the length of the key

Kasiski method (1852) - invented also by Charles Babbage (1853).

Basic observation If a subword of a plaintext is repeated at a distance that is a multiple of the length of the key, then the corresponding subwords of the cryptotext are the same.

Example, cryptotext:

CHRGQPWOEIRULYANDOSHCHRIZKEBUSNOFKYWROPDCHRKGAXBNRHROAKERBKSCHRIWK

Substring "CHR" occurs in positions 1, 21, 41, 66: expected keyword length is therefore 5.

Method. Determine the greatest common divisor of the distances between identical subwords (of length 3 or more) of the cryptotext.

Friedman method Let n_i be the number of occurrences of the **i**-*th* letter in the cryptotext.

Let **l** be the length of the keyword.

Let **n** be the length of the cryptotext.

Then it holds $l = \frac{0.027n}{(n-1)I - 0.038n + 0.065}$, $I = \sum_{i=1}^{26} \frac{n_i(n_i-1)}{n(n-1)}$

Once the length of the keyword is found it is easy to determine the key using the statistical (frequency analysis) method of analyzing monoalphabetic cryptosystems.

DERIVATION of the FRIEDMAN METHOD I

1. Let n_i be the number of occurrences of i-th alphabet symbol in a text of length n. The probability that if one selects a pair of symbols from the text, then they are the same is

$$I = \frac{\sum_{i=1}^{26} n_i(n_i - 1)}{n(n-1)} = \sum_{i=1}^{26} \frac{\binom{n_i}{2}}{\binom{n}{2}}$$

and it is called the index of coincidence.

DERIVATION of the FRIEDMAN METHOD I

1. Let n_i be the number of occurrences of i-th alphabet symbol in a text of length n. The probability that if one selects a pair of symbols from the text, then they are the same is

$$I = \frac{\sum_{i=1}^{26} n_i(n_i-1)}{n(n-1)} = \sum_{i=1}^{26} \frac{\binom{n_i}{2}}{\binom{n}{2}}$$

   and it is called the index of coincidence.

2. Let p_i be the probability that a randomly chosen symbol is the i-th symbol of the alphabet. The probability that two randomly chosen symbols are the same is

$$\sum_{i=1}^{26} p_i^2$$

   For English text one has

$$\sum_{i=1}^{26} p_i^2 = 0.065$$

   For randomly chosen text:

$$\sum_{i=1}^{26} p_i^2 = \sum_{i=1}^{26} \frac{1}{26^2} = 0.038$$

   Approximately

$$I = \sum_{i=1}^{26} p_i^2$$

Assume that a cryptotext is organized into l columns headed by the letters of the keyword

letters S_l	S_1	S_2	S_3	$\dots$	S_l
	x_1	x_2	x_3	$\dots$	X_l
	x_{l+1}	x_{l+2}	x_{l+3}		X_{2l}
	x_{2l+1}	x_{2l+2}	x_{2l+3}	$\dots$	x_{3l}
	.	.	.		.

First observation Each column is obtained using the CAESAR cryptosystem.
Probability that two randomly chosen letters are the same in

- the same column is 0.065.
- different columns is 0.038.

The number of pairs of letters in the same column: $\frac{l}{2} \cdot \frac{n}{l}(\frac{n}{l} - 1) = \frac{n(n-l)}{2l}$

The number of pairs of letters in different columns: $\frac{l(l-1)}{2} \cdot \frac{n^2}{l^2} = \frac{n^2(l-1)}{2l}$

The expected number A of pairs of equals letters is $A = \frac{n(n-l)}{2l} \cdot 0.065 + \frac{n^2(l-1)}{2l} \cdot 0.038$

Since $I = \frac{A}{\frac{n(n-1)}{2}} = \frac{1}{l(n-1)}[0.027n + l(0.038n - 0.065)]$

one gets the formula for l from the previous slide.

ONE-TIME PAD CRYPTOSYSTEM – Vernam's cipher

$$\text{Binary case:}\quad \left.\begin{array}{ll} \text{plaintext} & w \\ \text{key} & k \\ \text{cryptotext} & c \end{array}\right\}\ \text{are binary words of the same length}$$

Encryption: $c = w \oplus k$
Decryption: $w = c \oplus k$

ONE-TIME PAD CRYPTOSYSTEM – Vernam's cipher

Binary case: $\left.\begin{array}{ll} \text{plaintext} & w \\ \text{key} & k \\ \text{cryptotext} & c \end{array}\right\}$ are binary words of the same length

Encryption: $c = w \oplus k$

Decryption: $w = c \oplus k$

Example:

$$w = 101101011$$
$$k = 011011010$$
$$c = 110110001$$

ONE-TIME PAD CRYPTOSYSTEM – Vernam's cipher

Binary case: $\left.\begin{array}{ll} \text{plaintext} & w \\ \text{key} & k \\ \text{cryptotext} & c \end{array}\right\}$ are binary words of the same length

Encryption: $c = w \oplus k$
Decryption: $w = c \oplus k$
Example:

$$w = 101101011$$
$$k = 011011010$$
$$c = 110110001$$

What happens if the same key is used twice or 3 times for encryption?

ONE-TIME PAD CRYPTOSYSTEM – Vernam's cipher

Binary case: $\left.\begin{array}{ll}\text{plaintext} & w \\ \text{key} & k \\ \text{cryptotext} & c\end{array}\right\}$ are binary words of the same length

Encryption: $c = w \oplus k$

Decryption: $w = c \oplus k$

Example:

$$\begin{aligned} w &= 101101011 \\ k &= 011011010 \\ c &= 110110001 \end{aligned}$$

What happens if the same key is used twice or 3 times for encryption?

$$c_1 = w_1 \oplus k, c_2 = w_2 \oplus k, c_3 = w_3 \oplus k$$

$$\begin{aligned} c_1 \oplus c_2 &= w_1 \oplus w_2 \\ c_1 \oplus c_3 &= w_1 \oplus w_3 \\ c_2 \oplus c_3 &= w_2 \oplus w_3 \end{aligned}$$

PERFECT SECRET-KEY CRYPTOSYSTEMS

By Shannon, a cryptosystem is perfect if the knowledge of the cryptotext provides no information whatsoever about its plaintext (with the exception of its length).

It follows from Shannon's results that perfect secrecy is possible if the key-space is as large as the plaintext-space. In addition, a key has to be as long as plaintext and the same key should not be used twice.

PERFECT SECRET-KEY CRYPTOSYSTEMS

By Shannon, a cryptosystem is perfect if the knowledge of the cryptotext provides no information whatsoever about its plaintext (with the exception of its length).

It follows from Shannon's results that perfect secrecy is possible if the key-space is as large as the plaintext-space. In addition, a key has to be as long as plaintext and the same key should not be used twice.

An example of a perfect cryptosystem ONE-TIME PAD cryptosystem (Gilbert S. Vernam (1917) - AT&T + Major Joseph Mauborgne).

PERFECT SECRET-KEY CRYPTOSYSTEMS

By Shannon, a cryptosystem is perfect if the knowledge of the cryptotext provides no information whatsoever about its plaintext (with the exception of its length).

It follows from Shannon's results that perfect secrecy is possible if the key-space is as large as the plaintext-space. In addition, a key has to be as long as plaintext and the same key should not be used twice.

An example of a perfect cryptosystem ONE-TIME PAD cryptosystem (Gilbert S. Vernam (1917) - AT&T + Major Joseph Mauborgne).

If used with the English alphabet, it is simply a polyalphabetic substitution cryptosystem of VIGENERE with the key being a randomly chosen English word of the same length as the plaintext.

Proof of perfect secrecy: by the proper choice of the key any plaintext of the same length could provide the given cryptotext.

PERFECT SECRET-KEY CRYPTOSYSTEMS

By Shannon, a cryptosystem is perfect if the knowledge of the cryptotext provides no information whatsoever about its plaintext (with the exception of its length).

It follows from Shannon's results that perfect secrecy is possible if the key-space is as large as the plaintext-space. In addition, a key has to be as long as plaintext and the same key should not be used twice.

An example of a perfect cryptosystem ONE-TIME PAD cryptosystem (Gilbert S. Vernam (1917) - AT&T + Major Joseph Mauborgne).

If used with the English alphabet, it is simply a polyalphabetic substitution cryptosystem of VIGENERE with the key being a randomly chosen English word of the same length as the plaintext.

Proof of perfect secrecy: by the proper choice of the key any plaintext of the same length could provide the given cryptotext.

Did we gain something? The problem of secure communication of the plaintext got transformed to the problem of secure communication of the key of the same length.

PERFECT SECRET-KEY CRYPTOSYSTEMS

By Shannon, a cryptosystem is perfect if the knowledge of the cryptotext provides no information whatsoever about its plaintext (with the exception of its length).

It follows from Shannon's results that perfect secrecy is possible if the key-space is as large as the plaintext-space. In addition, a key has to be as long as plaintext and the same key should not be used twice.

An example of a perfect cryptosystem ONE-TIME PAD cryptosystem (Gilbert S. Vernam (1917) - AT&T + Major Joseph Mauborgne).

If used with the English alphabet, it is simply a polyalphabetic substitution cryptosystem of VIGENERE with the key being a randomly chosen English word of the same length as the plaintext.

Proof of perfect secrecy: by the proper choice of the key any plaintext of the same length could provide the given cryptotext.

Did we gain something? The problem of secure communication of the plaintext got transformed to the problem of secure communication of the key of the same length.

Yes:

1. ONE-TIME PAD cryptosystem is used in critical applications
2. It suggests an idea how to construct practically secure cryptosystems.

TRANSPOSITION CRYPTOSYSTEMS

The basic idea is very simple: permute the plaintext to get the cryptotext. Less clear it is how to specify and perform efficiently permutations.

One idea: choose n, write plaintext into rows, with n symbols in each row and then read it by columns to get cryptotext.

Example

I	N	J	E	D	E	M	M	E	N
S	C	H	E	N	G	E	S	I	C
H	T	E	S	T	E	H	T	S	E
I	N	E	G	E	S	C	H	I	C
H	T	E	T	O	J	E	O	N	O

Cryptotexts obtained by transpositions, called anagrams, were popular among scientists of 17th century. They were used also to encrypt scientific findings.

Newton wrote to Leibniz

$$a^7 c^2 d^2 e^{14} f^2 i^7 l^3 m^1 n^8 o^4 q^3 r^2 s^4 t^8 v^{12} x^1$$

what stands for: "data aequatione quodcumque fluentes quantitates involvente, fluxiones invenire et vice versa"

Example $a^2 cdef^3 g^2 i^2 jkmn^3 o^5 prs^2 t^2 u^3 z$

Solution:

KEYWORD CAESAR CRYPTOSYSTEM

Choose an integer $0 < k < 25$ and a string, called keyword, of length at most 25 with all letters different.

The keyword is then written bellow the English alphabet letters, beginning with the k-symbol, and the remaining letters are written in the alphabetic order and cyclically after the keyword.

KEYWORD CAESAR CRYPTOSYSTEM

Choose an integer $0 < k < 25$ and a string, called keyword, of length at most 25 with all letters different.

The keyword is then written bellow the English alphabet letters, beginning with the k-symbol, and the remaining letters are written in the alphabetic order and cyclically after the keyword.

Example: keyword: HOW MANY ELKS, $k = 8$

0								8																	
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
P	Q	R	T	U	V	X	Z	H	O	W	M	A	N	Y	E	L	K	S	B	C	D	F	G	I	J

KEYWORD CAESAR - Example I

Example Decrypt the following cryptotext encrypted using the KEYWORD CAESAR and determine the keyword and k

```
T   I V D   Z C R T I C   F Q N I Q   T U   T F
Q   X A V F C Z   F E Q X C   P C Q U C Z   W K
Q   F U V B C   F N R R T X T C I U A K   W T Y
D T U P   M C F E C X U   U V   U P C   B V A N H C
V R   U P C   F E Q X C   U P C   F U V B C
X V I U Q T I F   F U V I C F   N F N Q A A K
V I   U P C   U V E   U V   U Q G C   Q   F Q N I Q
W Q U P   T U   T F   Q A F V   I C X C F F Q M K
U P Q U   U P C   F U V B C   T F   E M V E C M A K
P C Q U C Z   Q I Z   U P Q U   K V N   P Q B C
U P C   R Q X T A T U K   V R   U P M V D T I Y
D Q U C M   V I   U P C   F U V I C F
```

KEYWORD CAESAR - Example II

Step 1. Make the frequency counts:

	Number		Number		Number
U	32	X	8	W	3
C	31	K	7	Y	2
Q	23	N	7	G	1
F	22	E	6	H	1
V	20	M	6	J	0
P	15	R	6	L	0
T	15	B	5	O	0
I	14	Z	5	S	0
A	8	D	4		
	180=74.69%		54=22.41%		7=2.90%

Step 1. Make the frequency counts:

	Number		Number		Number
U	32	X	8	W	3
C	31	K	7	Y	2
Q	23	N	7	G	1
F	22	E	6	H	1
V	20	M	6	J	0
P	15	R	6	L	0
T	15	B	5	O	0
I	14	Z	5	S	0
A	8	D	4		
	180=74.69%		54=22.41%		7=2.90%

Step 2. Cryptotext contains two one-letter words T and Q. They must be A and I. Since T occurs once and Q three times it is likely that T is I and Q is A.

The three letter word UPC occurs 7 times and all other 3-letter words occur only once. Hence

UPC is likely to be THE.

Let us now decrypt the remaining letters in the high frequency group: F,V,I

From the words TU, TF $\Rightarrow$ F=S
From UV $\Rightarrow$ V=O
From VI $\Rightarrow$ I=N

The result after the remaining guesses

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
L V E W P S K M N ? Y ? R U ? H E F ? I T O B C G D

Redundancy of natural languages is of the key importance for cryptanalysis.

Would all letters of a 26-symbol alphabet have the same probability, a character would carry $\lg 26 = 4.7$ bits of Information.

The estimated average amount of information carried per letter in a meaningful English text is 1.5 bits.

The unicity distance of a cryptosystem is the minimum number of cryptotext (number of letters) required to a computationally unlimited adversary to recover the unique encryption key.

Empirical evidence indicates that if any simple cryptosystem is applied to a meaningful English message, then about 25 cryptotext characters is enough for an experienced cryptanalyst to recover the plaintext.

ANAGRAMS – EXAMPLES

German:

IRI BRÄTER, GENF	Briefträgerin
FRANK PEKL, REGEN	...
PEER ASSSTIL, MELK	...
INGO DILMR, PEINE	...
EMIL REST, GERA	...
KARL SORDORT, PEINE	...

ANAGRAMS – EXAMPLES

German:

IRI BRÄTER, GENF	Briefträgerin
FRANK PEKL, REGEN	...
PEER ASSSTIL, MELK	...
INGO DILMR, PEINE	...
EMIL REST, GERA	...
KARL SORDORT, PEINE	...

English:

algorithms	logarithms
antagonist	stagnation
compressed	decompress
coordinate	decoration
creativity	reactivity
deductions	discounted
descriptor	predictors
impression	permission
introduces	reductions
procedures	reproduces

APPENDIX

STREAM CRYPTOSYSTEMS

Two basic types of cryptosystems are:

- Block cryptosystems (Hill cryptosystem,. . .) – they are used to encrypt simultaneously blocks of plaintext.
- Stream cryptosystems (CAESAR, ONE-TIME PAD,. . .) – they encrypt plaintext letter by letter, or block by block, using an encryption that may vary during the encryption process.

Stream cryptosystems are more appropriate in some applications (telecommunication), usually are simpler to implement (also in hardware), usually are faster and usually have no error propagation (what is of importance when transmission errors are highly probable).

Two basic types of stream cryptosystems: secret key cryptosystems (ONE-TIME PAD) and public-key cryptosystems (Blum-Goldwasser)

Block versus stream cryptosystems

In block cryptosystems the same key is used to encrypt arbitrarily long plaintext – block by block - (after dividing each long plaintext w into a sequence of subplaintexts (blocks) $w_1 w_2 w_3$).

In stream cryptosystems each block is encrypted using a different key

Block versus stream cryptosystems

In block cryptosystems the same key is used to encrypt arbitrarily long plaintext – block by block - (after dividing each long plaintext w into a sequence of subplaintexts (blocks) $w_1 w_2 w_3$).

In stream cryptosystems each block is encrypted using a different key

- The fixed key k is used to encrypt all blocks. In such a case the resulting cryptotext has the form

$$c = c_1 c_2 c_3 \ldots = e_k(w_1) e_k(w_2) e_k(w_3) \ldots$$

Block versus stream cryptosystems

In block cryptosystems the same key is used to encrypt arbitrarily long plaintext – block by block - (after dividing each long plaintext w into a sequence of subplaintexts (blocks) $w_1 w_2 w_3$).

In stream cryptosystems each block is encrypted using a different key

- The fixed key k is used to encrypt all blocks. In such a case the resulting cryptotext has the form

$$c = c_1 c_2 c_3 \ldots = e_k(w_1) e_k(w_2) e_k(w_3) \ldots$$

- A stream of keys is used to encrypt subplaintexts. The basic idea is to generate a key-stream $K = k_1, k_2, k_3, \ldots$ and then to compute the cryptotext as follows

$$c = c_1 c_2 c_3 \ldots = e_{k1}(w_1) e_{k2}(w_2) e_{k3}(w_3).$$

CRYPTOSYSTEMS WITH STREAMS OF KEYS

Various techniques are used to compute a sequence of keys. For example, given a key k

$$k_i = f_i(k, k_1, k_2, \ldots, k_{i-1})$$

In such a case encryption and decryption processes generate the following sequences:

Encryption: To encrypt the plaintext $w_1 w_2 w_3 \ldots$ the sequence

$$k_1, c_1, k_2, c_2, k_3, c_3, \ldots$$

of keys and sub-cryptotexts is computed.

CRYPTOSYSTEMS WITH STREAMS OF KEYS

Various techniques are used to compute a sequence of keys. For example, given a key k

$$k_i = f_i(k, k_1, k_2, \ldots, k_{i-1})$$

In such a case encryption and decryption processes generate the following sequences:

Encryption: To encrypt the plaintext $w_1 w_2 w_3 \ldots$ the sequence

$$k_1, c_1, k_2, c_2, k_3, c_3, \ldots$$

of keys and sub-cryptotexts is computed.

Decryption: To decrypt the cryptotext $c_1 c_2 c_3 \ldots$ the sequence

$$k_1, w_1, k_2, w_2, k_3, w_3, \ldots$$

of keys and subplaintexts is computed.

EXAMPLES

A keystream is called synchronous if it is independent of the plaintext.

KEYWORD VIGENERE cryptosystem can be seen as an example of a synchronous keystream cryptosystem.

Another type of the binary keystream cryptosystem is specified by an initial sequence of keys $k_1, k_2, k_3 \dots k_m$

and an initial sequence of binary constants $b_1, b_2, b_3 \dots b_{m-1}$

and the remaining keys are computed using the rule

$$k_{i+m} = \sum_{j=0}^{m-1} b_j k_{i+j} \bmod 2$$

A keystream is called periodic with period p if $k_{i+p} = k_i$ for all i.

EXAMPLES

A keystream is called synchronous if it is independent of the plaintext.

KEYWORD VIGENERE cryptosystem can be seen as an example of a synchronous keystream cryptosystem.

Another type of the binary keystream cryptosystem is specified by an initial sequence of keys $k_1, k_2, k_3 \ldots k_m$

and an initial sequence of binary constants $b_1, b_2, b_3 \ldots b_{m-1}$

and the remaining keys are computed using the rule

$$k_{i+m} = \sum_{j=0}^{m-1} b_j k_{i+j} \bmod 2$$

A keystream is called periodic with period p if $k_{i+p} = k_i$ for all i.

Example Let the keystream be generated by the rule

$$k_{i+4} = k_i \oplus k_{i+1}$$

If the initial sequence of keys is (1,0,0,0), then we get the following keystream:

1,0,0,0,1,0,0,1,1,0,1,0 1,1,1, . . .

of period 15.

Let **P, K** and **C** be sets of plaintexts, keys and cryptotexts.
Let $p_K(k)$ be the probability that the key k is chosen from **K** and let a priory probability that plaintext w is chosen be $p_p(w)$.
If for a key $k \in K, C(k) = \{e_k(w)|w \in P\}$, then for the probability $P_C(y)$ that c is the cryptotext that is transmitted it holds

$$p_c(c) = \sum_{\{k|c \in C(k)\}} p_K(k)p_P(d_k(c)).$$

For the conditional probability $p_c(c|w)$ that c is the cryptotext if w is the plaintext it holds

$$p_c(c|w) = \sum_{\{k|w=d_k(c)\}} p_K(k).$$

Using Bayes' conditional probability formula $p(y)p(x|y) = p(x)p(y|x)$ we get for probability $p_P(w|c)$ that w is the plaintext if c is the cryptotext the expression

$$p_P = \frac{P_P(w) \sum_{\{k|w=d_k(c)\}} p_K(k)}{\sum_{\{k|c \in C(K)\}} p_K(k)p_P(d_K(c))}.$$

PERFECT SECRECY - BASIC RESULTS

Definition A cryptosystem has perfect secrecy if

$$p_P(w|c) = p_P(w) \text{ for all } w \in P \text{ and } c \in C.$$

(That is, the a posteriori probability that the plaintext is w,given that the cryptotext is c is obtained, is the same as a priori probability that the plaintext is w.)

Example CAESAR cryptosystem has perfect secrecy if any of the 26 keys is used with the same probability to encode any symbol of the plaintext.

Proof Exercise.

An analysis of perfect secrecy: The condition $p_P(w|c) = p_P(w)$ is for all $w \in P$ and $c \in C$ equivalent to the condition $p_C(c|w) = p_C(c)$.

Let us now assume that $p_C(c) > 0$ for all $c \in C$.

Fix $w \in P$. For each $c \in C$ we have $p_C(c|w) = p_C(c) > 0$. Hence, for each $c \in C$ there must exist at least one key k such that $e_k(w) = c$. Consequently, $|K| \geq |C| \geq |P|$.

In a special case $|K| = |C| = |P|$, the following nice characterization of the perfect secrecy can be obtained:

Theorem A cryptosystem in which $|P| = |K| = |C|$ provides perfect secrecy if and only if every key is used with the same probability and for every $w \in P$ and every $c \in C$ there is a unique key k such that $e_k(w) = c$.

Proof Exercise.

PRODUCT CRYPTOSYSTEMS

A cryptosystem $S = (P, K, C, e, d)$ with the sets of plaintexts P, keys K and cryptotexts C and encryption (decryption) algorithms $e(d)$ is called **endomorphic** if $P = C$.

If $S_1 = (P, K_1, P, e^{(1)}, d^{(1)})andS_2 = (P, K_2, P, e^{(2)}, d^{(2)})$ are endomorphic cryptosystems, then the **product cryptosystem** is

$$S_1 \otimes S_2 = (P, K_1 \otimes K_2, P, e, d),$$

where encryption is performed by the procedure

$$e_{(k1,k2)}(w) = e_{k2}(e_{k1}(w))$$

and decryption by the procedure

$$d_{(k1,k2)}(c) = d_{k1}(d_{k2}(c)).$$

PRODUCT CRYPTOSYSTEMS

A cryptosystem $S = (P, K, C, e, d)$ with the sets of plaintexts P, keys K and cryptotexts C and encryption (decryption) algorithms $e(d)$ is called **endomorphic** if $P = C$.

If $S_1 = (P, K_1, P, e^{(1)}, d^{(1)})andS_2 = (P, K_2, P, e^{(2)}, d^{(2)})$ are endomorphic cryptosystems, then the **product cryptosystem** is

$$S_1 \otimes S_2 = (P, K_1 \otimes K_2, P, e, d),$$

where encryption is performed by the procedure

$$e_{(k1,k2)}(w) = e_{k2}(e_{k1}(w))$$

and decryption by the procedure

$$d_{(k1,k2)}(c) = d_{k1}(d_{k2}(c)).$$

Example (Multiplicative cryptosystem):

Encryption: $e_a(w) = aw \bmod p$; decryption: $d_a(c) = a^{-1}c \bmod 26$.

If M denote the multiplicative cryptosystem, then clearly CAESAR $\times$ M is actually the AFFINE cryptosystem.

PRODUCT CRYPTOSYSTEMS

A cryptosystem $S = (P, K, C, e, d)$ with the sets of plaintexts P, keys K and cryptotexts C and encryption (decryption) algorithms $e(d)$ is called **endomorphic** if $P = C$.

If $S_1 = (P, K_1, P, e^{(1)}, d^{(1)})andS_2 = (P, K_2, P, e^{(2)}, d^{(2)})$ are endomorphic cryptosystems, then the **product cryptosystem** is

$$S_1 \otimes S_2 = (P, K_1 \otimes K_2, P, e, d),$$

where encryption is performed by the procedure

$$e_{(k1,k2)}(w) = e_{k2}(e_{k1}(w))$$

and decryption by the procedure

$$d_{(k1,k2)}(c) = d_{k1}(d_{k2}(c)).$$

Example (Multiplicative cryptosystem):

Encryption: $e_a(w) = aw \bmod p$; decryption: $d_a(c) = a^{-1}c \bmod 26$.

If M denote the multiplicative cryptosystem, then clearly CAESAR $\times$ M is actually the AFFINE cryptosystem.

Exercise Show that also M $\otimes$ CAESAR is actually the AFFINE cryptosystem.

Two cryptosystems S_1 and S_2 are called **commutative** if $S_1 \otimes S_2 = S_2 \otimes S_1$.

A cryptosystem S is called **idempotent** if $S \otimes S = S$.

Part V

Public-key cryptosystems, I. Key exchange, knapsack, RSA

Rapidly increasing needs for flexible and secure transmission of information require to use new cryptographic methods.

The main disadvantage of the classical (symmetric) cryptography is the need to send a (long) key through a super secure channel before sending the message itself.

Rapidly increasing needs for flexible and secure transmission of information require to use new cryptographic methods.

The main disadvantage of the classical (symmetric) cryptography is the need to send a (long) key through a super secure channel before sending the message itself.

In the classical or secret-key (symmetric) cryptography both sender and receiver share the same secret key.

Rapidly increasing needs for flexible and secure transmission of information require to use new cryptographic methods.

The main disadvantage of the classical (symmetric) cryptography is the need to send a (long) key through a super secure channel before sending the message itself.

In the classical or secret-key (symmetric) cryptography both sender and receiver share the same secret key.

In the public-key (asymmetric) cryptography there are two different keys:

a public encryption key (at the sender side)

and

a private (secret) decryption key (at the receiver side).

BASIC IDEA - EXAMPLE

Basic idea: If it is infeasible from the knowledge of an encryption algorithm e_k to construct the corresponding description algorithm d_k, then e_k can be made public.

Toy example: (Telephone directory encryption)

Start: Each user **U** makes public a unique telephone directory td_U to encrypt messages for **U** and **U** is the only user to have an inverse telephone directory itd_U.

Encryption: Each letter **X** of a plaintext **w** is replaced, using the telephone directory td_U of the intended receiver **U**, by the telephone number of a person whose name starts with letter **X**.

Decryption: easy for U_k, with the inverse telephone directory, infeasible for others.

BASIC IDEA - EXAMPLE

Basic idea: If it is infeasible from the knowledge of an encryption algorithm e_k to construct the corresponding description algorithm d_k, then e_k can be made public.

Toy example: (Telephone directory encryption)

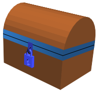
Start: Each user **U** makes public a unique telephone directory td_U to encrypt messages for **U** and **U** is the only user to have an inverse telephone directory itd_U.

Encryption: Each letter **X** of a plaintext **w** is replaced, using the telephone directory td_U of the intended receiver **U**, by the telephone number of a person whose name starts with letter **X**.

Decryption: easy for U_k, with the inverse telephone directory, infeasible for others.

Analogy between secret and public-key cryptography:

Secret-key cryptography *1.* Put the message into a box, lock it with a padlock and send the box. *2.* Send the key by a secure channel.



Public-key cryptography Open padlocks, for each user different ones, are freely available. Only legitimate user has key from his padlocks. *Transmission*: Put the message into the box of the intended receiver, close the padlock and send the box.

PUBLIC ESTABLISHMENT of SECRET KEYS

Main problem of the secret-key cryptography: a need to make a secure distribution (establishment) of secret keys ahead of transmissions.

Diffie+Hellman solved this problem in 1976 by designing a protocol for secure key establishment (distribution) over public channels.

PUBLIC ESTABLISHMENT of SECRET KEYS

Main problem of the secret-key cryptography: a need to make a secure distribution (establishment) of secret keys ahead of transmissions.

Diffie+Hellman solved this problem in 1976 by designing a protocol for secure key establishment (distribution) over public channels.

Diffie-Hellman Protocol: If two parties, Alice and Bob, want to create a common secret key, then they first agree, somehow, on a large prime p and a q<p of large order in Z_p^* and then they perform, through a public channel, the following activities.

- Alice chooses, randomly, a large $1 \leq x < p-1$ and computes

$$X = q^x \bmod p.$$

Main problem of the secret-key cryptography: a need to make a secure distribution (establishment) of secret keys ahead of transmissions.

Diffie+Hellman solved this problem in 1976 by designing a protocol for secure key establishment (distribution) over public channels.

Diffie-Hellman Protocol: If two parties, Alice and Bob, want to create a common secret key, then they first agree, somehow, on a large prime p and a q<p of large order in Z_p^* and then they perform, through a public channel, the following activities.

- Alice chooses, randomly, a large $1 \leq x < p-1$ and computes

$$X = q^x \bmod p.$$

- Bob also chooses, again randomly, a large $1 \leq y < p-1$ and computes

$$Y = q^y \bmod p.$$

Main problem of the secret-key cryptography: a need to make a secure distribution (establishment) of secret keys ahead of transmissions.

Diffie+Hellman solved this problem in 1976 by designing a protocol for secure key establishment (distribution) over public channels.

Diffie-Hellman Protocol: If two parties, Alice and Bob, want to create a common secret key, then they first agree, somehow, on a large prime p and a q<p of large order in Z_p^* and then they perform, through a public channel, the following activities.

- Alice chooses, randomly, a large $1 \leq x < p-1$ and computes
$$X = q^x \bmod p.$$
- Bob also chooses, again randomly, a large $1 \leq y < p-1$ and computes
$$Y = q^y \bmod p.$$
- Alice and Bob exchange **X** and **Y**, through a public channel, but keep **x, y** secret.

Main problem of the secret-key cryptography: a need to make a secure distribution (establishment) of secret keys ahead of transmissions.

Diffie+Hellman solved this problem in 1976 by designing a protocol for secure key establishment (distribution) over public channels.

Diffie-Hellman Protocol: If two parties, Alice and Bob, want to create a common secret key, then they first agree, somehow, on a large prime p and a q<p of large order in Z_p^* and then they perform, through a public channel, the following activities.

- Alice chooses, randomly, a large $1 \leq x < p - 1$ and computes
$$X = q^x \bmod p.$$
- Bob also chooses, again randomly, a large $1 \leq y < p - 1$ and computes
$$Y = q^y \bmod p.$$
- Alice and Bob exchange **X** and **Y**, through a public channel, but keep **x, y** secret.
- Alice computes $Y^x \bmod p$ and Bob computes $X^y \bmod p$ and then each of them has the key
$$K = q^{xy} \bmod p.$$

PUBLIC ESTABLISHMENT of SECRET KEYS

Main problem of the secret-key cryptography: a need to make a secure distribution (establishment) of secret keys ahead of transmissions.

Diffie+Hellman solved this problem in 1976 by designing a protocol for secure key establishment (distribution) over public channels.

Diffie-Hellman Protocol: If two parties, Alice and Bob, want to create a common secret key, then they first agree, somehow, on a large prime p and a q<p of large order in Z_p^* and then they perform, through a public channel, the following activities.

- Alice chooses, randomly, a large $1 \leq x < p-1$ and computes
$$X = q^x \bmod p.$$
- Bob also chooses, again randomly, a large $1 \leq y < p-1$ and computes
$$Y = q^y \bmod p.$$
- Alice and Bob exchange **X** and **Y**, through a public channel, but keep **x, y** secret.
- Alice computes $Y^x \bmod p$ and Bob computes $X^y \bmod p$ and then each of them has the key
$$K = q^{xy} \bmod p.$$

PUBLIC ESTABLISHMENT of SECRET KEYS

Main problem of the secret-key cryptography: a need to make a secure distribution (establishment) of secret keys ahead of transmissions.

Diffie+Hellman solved this problem in 1976 by designing a protocol for secure key establishment (distribution) over public channels.

Diffie-Hellman Protocol: If two parties, Alice and Bob, want to create a common secret key, then they first agree, somehow, on a large prime p and a q<p of large order in Z_p^* and then they perform, through a public channel, the following activities.

- Alice chooses, randomly, a large $1 \leq x < p-1$ and computes

$$X = q^x \bmod p.$$

- Bob also chooses, again randomly, a large $1 \leq y < p-1$ and computes

$$Y = q^y \bmod p.$$

- Alice and Bob exchange **X** and **Y**, through a public channel, but keep **x, y** secret.
- Alice computes $Y^x \bmod p$ and Bob computes $X^y \bmod p$ and then each of them has the key

$$K = q^{xy} \bmod p.$$

An eavesdropper seems to need, in order to determine x from **X, q, p** and y from **Y, q, p**, a capability to compute discrete logarithms, or to compute q^{xy} from q^x and q^y, what is believed to be infeasible.

KEY DISTRIBUTION / AGREEMENT

One should distinguish between key distribution and key agreement.

- Key distribution is a mechanism whereby one party chooses a secret key and then transmits it to another party or parties.
- Key agreement is a protocol whereby two (or more) parties jointly establish a secret key by communication over a public channel.

The objective of key distribution or key agreement protocols is that, at the end of the protocols, the two parties involved both have possession of the same key k, and the value of k is not known (at all) to any other party.

MAN-IN-THE-MIDDLE ATTACKS

The following attack, by a man-in-the-middle, is possible against the Diffie-Hellman key establishment protocol.

The following attack, by a man-in-the-middle, is possible against the Diffie-Hellman key establishment protocol.

1. Eve chooses an exponent z.

MAN-IN-THE-MIDDLE ATTACKS

The following attack, by a man-in-the-middle, is possible against the Diffie-Hellman key establishment protocol.

1. Eve chooses an exponent z.
2. Eve intercepts q^x and q^y.

MAN-IN-THE-MIDDLE ATTACKS

The following attack, by a man-in-the-middle, is possible against the Diffie-Hellman key establishment protocol.

1. Eve chooses an exponent z.
2. Eve intercepts q^x and q^y.
3. Eve sends q^z to both Alice and Bob. (After that Alice believes she has received q^y and Bob believes he has received q^x.)

The following attack, by a man-in-the-middle, is possible against the Diffie-Hellman key establishment protocol.

1. Eve chooses an exponent z.
2. Eve intercepts q^x and q^y.
3. Eve sends q^z to both Alice and Bob. (After that Alice believes she has received q^y and Bob believes he has received q^x.)
4. Eve computes $K_A = q^{xz} \pmod{p}$ and $K_B = q^{yz} \pmod{p}$.
   Alice, not realizing that Eve is in the middle, also computes K_A and
   Bob, not realizing that Eve is in the middle, also computes K_B.

MAN-IN-THE-MIDDLE ATTACKS

The following attack, by a man-in-the-middle, is possible against the Diffie-Hellman key establishment protocol.

1. Eve chooses an exponent z.
2. Eve intercepts q^x and q^y.
3. Eve sends q^z to both Alice and Bob. (After that Alice believes she has received q^y and Bob believes he has received q^x.)
4. Eve computes $K_A = q^{xz} \pmod{p}$ and $K_B = q^{yz} \pmod{p}$.
   Alice, not realizing that Eve is in the middle, also computes K_A and
   Bob, not realizing that Eve is in the middle, also computes K_B.
5. When Alice sends a message to Bob, encrypted with K_A, Eve intercepts it, decrypts it, then encrypts it with K_B and sends it to Bob.

MAN-IN-THE-MIDDLE ATTACKS

The following attack, by a man-in-the-middle, is possible against the Diffie-Hellman key establishment protocol.

1. Eve chooses an exponent z.
2. Eve intercepts q^x and q^y.
3. Eve sends q^z to both Alice and Bob. (After that Alice believes she has received q^y and Bob believes he has received q^x.)
4. Eve computes $K_A = q^{xz}$ (mod p) and $K_B = q^{yz}$ (mod p).
   Alice, not realizing that Eve is in the middle, also computes K_A and
   Bob, not realizing that Eve is in the middle, also computes K_B.
5. When Alice sends a message to Bob, encrypted with K_A, Eve intercepts it, decrypts it, then encrypts it with K_B and sends it to Bob.
6. Bob decrypts the message with K_B and obtains the message. At this point he has no reason to think that communication was insecure.

MAN-IN-THE-MIDDLE ATTACKS

The following attack, by a man-in-the-middle, is possible against the Diffie-Hellman key establishment protocol.

1. Eve chooses an exponent z.
2. Eve intercepts q^x and q^y.
3. Eve sends q^z to both Alice and Bob. (After that Alice believes she has received q^y and Bob believes he has received q^x.)
4. Eve computes $K_A = q^{xz} \pmod p$ and $K_B = q^{yz} \pmod p$.
   Alice, not realizing that Eve is in the middle, also computes K_A and
   Bob, not realizing that Eve is in the middle, also computes K_B.
5. When Alice sends a message to Bob, encrypted with K_A, Eve intercepts it, decrypts it, then encrypts it with K_B and sends it to Bob.
6. Bob decrypts the message with K_B and obtains the message. At this point he has no reason to think that communication was insecure.
7. Meanwhile, Eve enjoys reading Alice's message.

BLOOM's KEY PRE-DISTRIBUTION PROTOCOL

allows a trusted authority (Trent - TA) to distribute secret keys to $\frac{n(n-1)}{2}$ pairs of n users.

Let a large prime $p > n$ be publicly known. Steps of the protocol:

1. Each user U in the network is assigned, by Trent, a unique public number $r_U < p$.

allows a trusted authority (Trent - TA) to distribute secret keys to $\frac{n(n-1)}{2}$ pairs of n users.

Let a large prime $p > n$ be publicly known. Steps of the protocol:

1. Each user U in the network is assigned, by Trent, a unique public number $r_U < p$.
2. Trent chooses three random numbers a, b and c, smaller than p.

BLOOM's KEY PRE-DISTRIBUTION PROTOCOL

allows a trusted authority (Trent - TA) to distribute secret keys to $\frac{n(n-1)}{2}$ pairs of n users.

Let a large prime $p > n$ be publicly known. Steps of the protocol:

1. Each user U in the network is assigned, by Trent, a unique public number $r_U < p$.
2. Trent chooses three random numbers a, b and c, smaller than p.
3. For each user U, Trent calculates two numbers

$$a_U = (a + br_U) \bmod p, \qquad b_U = (b + cr_U) \bmod p$$

   and sends them via his secure channel to U.

allows a trusted authority (Trent - TA) to distribute secret keys to $\frac{n(n-1)}{2}$ pairs of n users.

Let a large prime $p > n$ be publicly known. Steps of the protocol:

1. Each user U in the network is assigned, by Trent, a unique public number $r_U < p$.
2. Trent chooses three random numbers a, b and c, smaller than p.
3. For each user U, Trent calculates two numbers

$$a_U = (a + br_U) \bmod p, \qquad b_U = (b + cr_U) \bmod p$$

   and sends them via his secure channel to U.

4. Each user U creates the polynomial

$$g_U(x) = a_U + b_U(x).$$

BLOOM's KEY PRE-DISTRIBUTION PROTOCOL

allows a trusted authority (Trent - TA) to distribute secret keys to $\frac{n(n-1)}{2}$ pairs of n users.

Let a large prime $p > n$ be publicly known. Steps of the protocol:

1. Each user U in the network is assigned, by Trent, a unique public number $r_U < p$.
2. Trent chooses three random numbers a, b and c, smaller than p.
3. For each user U, Trent calculates two numbers

$$a_U = (a + br_U) \bmod p, \qquad b_U = (b + cr_U) \bmod p$$

   and sends them via his secure channel to U.
4. Each user U creates the polynomial

$$g_U(x) = a_U + b_U(x).$$

5. If Alice (A) wants to send a message to Bob (B), then Alice computes her key $K_{AB} = g_A(r_B)$ and Bob computes his key $K_{BA} = g_B(r_A)$.

BLOOM's KEY PRE-DISTRIBUTION PROTOCOL

allows a trusted authority (Trent - TA) to distribute secret keys to $\frac{n(n-1)}{2}$ pairs of n users.

Let a large prime $p > n$ be publicly known. Steps of the protocol:

1. Each user U in the network is assigned, by Trent, a unique public number $r_U < p$.
2. Trent chooses three random numbers a, b and c, smaller than p.
3. For each user U, Trent calculates two numbers

$$a_U = (a + br_U) \bmod p, \qquad b_U = (b + cr_U) \bmod p$$

   and sends them via his secure channel to U.

4. Each user U creates the polynomial

$$g_U(x) = a_U + b_U(x).$$

5. If Alice (A) wants to send a message to Bob (B), then Alice computes her key $K_{AB} = g_A(r_B)$ and Bob computes his key $K_{BA} = g_B(r_A)$.
6. It is easy to see that $K_{AB} = K_{BA}$ and therefore Alice and Bob can now use their (identical) keys to communicate using some secret-key cryptosystem.

and without any need for secret key distribution

(Shamir's "no-key algorithm")

Basic assumption: Each user X has its own

secret encryption function e_X

secret decryption function d_X

and all these functions commute (to form a commutative cryptosystem).

and without any need for secret key distribution

(Shamir's "no-key algorithm")

Basic assumption: Each user X has its own

secret encryption function e_X

secret decryption function d_X

and all these functions commute (to form a commutative cryptosystem).

Communication protocol

with which Alice can send a message w to Bob.

1. Alice sends $e_A(w)$ to Bob
2. Bob sends $e_B(e_A(w))$ to Alice
3. Alice sends $d_A(e_B(e_A(w))) = e_B(w)$ to Bob
4. Bob performs the decryption to get $d_B(e_B(w)) = w$.

and without any need for secret key distribution

(Shamir's "no-key algorithm")

Basic assumption: Each user X has its own

secret encryption function e_X

secret decryption function d_X

and all these functions commute (to form a commutative cryptosystem).

Communication protocol

with which Alice can send a message w to Bob.

1. Alice sends $e_A(w)$ to Bob
2. Bob sends $e_B(e_A(w))$ to Alice
3. Alice sends $d_A(e_B(e_A(w))) = e_B(w)$ to Bob
4. Bob performs the decryption to get $d_B(e_B(w)) = w$.

Disadvantage: 3 communications are needed (in such a context 3 is a much too large number).

Advantage: A perfect protocol for distribution of secret keys.

Modern cryptography uses such encryption methods that no "enemy" can have enough computational power and time to do decryption (even those capable to use thousands of supercomputers during tens of years for encryption).

Modern cryptography is based on negative and positive results of complexity theory – on the fact that for some algorithm problems no efficient algorithm seem to exists, surprisingly, and for some "small" modifications of these problems, surprisingly, simple, fast and good (randomized) algorithms do exist. Examples:

Modern cryptography uses such encryption methods that no "enemy" can have enough computational power and time to do decryption (even those capable to use thousands of supercomputers during tens of years for encryption).

Modern cryptography is based on negative and positive results of complexity theory – on the fact that for some algorithm problems no efficient algorithm seem to exists, surprisingly, and for some "small" modifications of these problems, surprisingly, simple, fast and good (randomized) algorithms do exist. Examples:

Integer factorization: Given $n(=pq)$, it is, in general, unfeasible, to find p, q.

There is a list of "most wanted to factor integers". Top recent successes, using thousands of computers for months.

(*) Factorization of $2^{2^9}+1$ with 155 digits (1996)

(**) Factorization of a "typical" 155-digits integer (1999)

Modern cryptography uses such encryption methods that no "enemy" can have enough computational power and time to do decryption (even those capable to use thousands of supercomputers during tens of years for encryption).

Modern cryptography is based on negative and positive results of complexity theory – on the fact that for some algorithm problems no efficient algorithm seem to exists, surprisingly, and for some "small" modifications of these problems, surprisingly, simple, fast and good (randomized) algorithms do exist. Examples:

Integer factorization: Given $n(=pq)$, it is, in general, unfeasible, to find p, q.

There is a list of "most wanted to factor integers". Top recent successes, using thousands of computers for months.

(*) Factorization of $2^{2^9}+1$ with 155 digits (1996)

(**) Factorization of a "typical" 155-digits integer (1999)

Primes recognition: Is a given n a prime? – fast randomized algorithms exist (1977). The existence of polynomial deterministic algorithms has been shown only in 2002

Discrete logarithm problem: Given x, y, n, determine integer a such that $y \equiv x^a (\mathrm{mod}\ n)$ – infeasible in general.

Discrete logarithm problem: Given x, y, n, determine integer a such that $y \equiv x^a (\mathrm{mod}\ n)$ – infeasible in general.

Discrete square root problem: Given integers y, n, compute an integer x such that $y \equiv x^2 (\mathrm{mod}\ n)$ – infeasible in general, easy if factorization of n is known

Discrete logarithm problem: Given x, y, n, determine integer a such that $y \equiv x^a (\mathrm{mod}\ n)$ – infeasible in general.

Discrete square root problem: Given integers y, n, compute an integer x such that $y \equiv x^2 (\mathrm{mod}\ n)$ – infeasible in general, easy if factorization of n is known

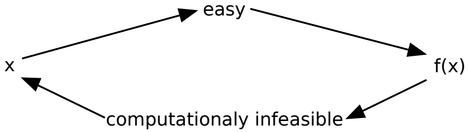
Knapsack problem: Given a (knapsack - integer) vector $X = (x_1, \ldots, x_n)$ and a (integer capacity) c, find a binary vector $(b_1, \ldots, b_n)$ such that

$$\sum_{i=1}^{n} b_i x_i = c.$$

Problem is NP-hard in general, but easy if $x_i > \sum_{j=1}^{i-1} x_j, 1 < i \leq n$.

Informally, a function $F : N \rightarrow N$ is said to be one-way function if it is easily computable - in polynomial time - but any computation of its inverse is infeasible.

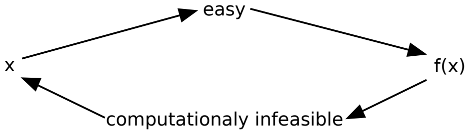
A one-way permutation is a 1-1 one-way function.



ONE-WAY FUNCTIONS

Informally, a function $F : N \rightarrow N$ is said to be one-way function if it is easily computable - in polynomial time - but any computation of its inverse is infeasible.

A one-way permutation is a 1-1 one-way function.



A more formal approach

Definition A function $f : \{0,1\}^* \rightarrow \{0,1\}^*$ is called a strongly one-way function if the following conditions are satisfied:

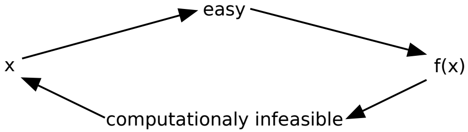
1. f can be computed in polynomial time;
2. there are $c, \varepsilon > 0$ such that $|x|^{\varepsilon} \leq |f(x)| \leq |x|^{c}$;
3. for every randomized polynomial time algorithm A, and any constant $c > 0$, there exists an n_c such that for $n > n_c$

$$P_r(A(f(x)) \in f^{-1}(f(x))) < \frac{1}{n^c}.$$

ONE-WAY FUNCTIONS

Informally, a function $F : N \rightarrow N$ is said to be one-way function if it is easily computable - in polynomial time - but any computation of its inverse is infeasible.

A one-way permutation is a 1-1 one-way function.



A more formal approach

Definition A function $f : \{0,1\}^* \rightarrow \{0,1\}^*$ is called a strongly one-way function if the following conditions are satisfied:

1. f can be computed in polynomial time;
2. there are $c, \varepsilon > 0$ such that $|x|^{\varepsilon} \leq |f(x)| \leq |x|^{c}$;
3. for every randomized polynomial time algorithm A, and any constant $c > 0$, there exists an n_c such that for $n > n_c$

$$P_r(A(f(x)) \in f^{-1}(f(x))) < \frac{1}{n^c}.$$

Candidates:
Modular exponentiation: $f(x) = a^x \bmod n$
Modular squaring $f(x) = x^2 \bmod n$, n – a Blum integer
Prime number multiplication $f(p,q) = pq$.

TRAPDOOR ONE-WAY FUNCTIONS

The key concept for design of public-key cryptosystems is that of trapdoor one-way functions.

A function $f : X \rightarrow Y$ is trapdoor one-way function

- if f and its inverse can be computed efficiently,
- yet even the complete knowledge of the algorithm to compute f does not make it feasible to determine a polynomial time algorithm to compute the inverse of f.

TRAPDOOR ONE-WAY FUNCTIONS

The key concept for design of public-key cryptosystems is that of trapdoor one-way functions.

A function $f : X \rightarrow Y$ is trapdoor one-way function

- if f and its inverse can be computed efficiently,
- yet even the complete knowledge of the algorithm to compute f does not make it feasible to determine a polynomial time algorithm to compute the inverse of f.

A candidate: modular squaring with a fixed modulus.

- computation of discrete square roots is unfeasible in general, but quite easy if the decomposition of the modulus into primes is known.

A way to design a trapdoor one-way function is to transform an easy case of a hard (one-way) function to a hard-looking case of such a function, that can be, however, solved easily by those knowing how the above transformation was performed.

A naive solution is to keep in computer a file with entries as

login CLINTON password BUSH,

that is with logins and their passwords. This is not sufficiently safe.

EXAMPLE - COMPUTER PASSWORDS

A naive solution is to keep in computer a file with entries as

login CLINTON password BUSH,

that is with logins and their passwords. This is not sufficiently safe.

A more safe method is to keep in the computer a file with entries as

login CLINTON password BUSH one-way function f_c

EXAMPLE - COMPUTER PASSWORDS

A naive solution is to keep in computer a file with entries as

login CLINTON password BUSH,

that is with logins and their passwords. This is not sufficiently safe.

A more safe method is to keep in the computer a file with entries as

login CLINTON password BUSH one-way function f_c

The idea is that BUSH is a "public" password and CLINTON is the only one that knows a "secret" password, say MADONNA, such that

$$f_c(\text{MADONNA}) = \text{BUSH}$$

LAMPORT's ONE-TIME PASSWORDS

One-way functions can be used to create a sequence of passwords:

1. Alice chooses a random w and computes, using a one-way function h, a sequence of passwords

$$w, h(w), h(h(w)), \ldots, h^n(w)$$

2. Alice then transfers securely "the initial secret" $w_0 = h^n(w)$ to Bob.
3. The i-th authentication, $0 < i < n+1$, is performed as follows:

- - - - - - - Alice sends $w_i = h^{n-i}(w)$ to Bob for I = 1, 2,. . . ,n-1

- - - - - - - Bob checks whether $w_{i-1} = h(w_i)$.

When the number of identifications reaches n, a new w has to be chosen.

GENERAL KNAPSACK PROBLEM – UNFEASIBLE

KNAPSACK PROBLEM: Given an integer-vector $X = (x_1, \ldots, x_n)$ and an integer c. Determine a binary vector $B = (b_1, \ldots, b_n)$ (if it exists) such that $XB^T = c$.

GENERAL KNAPSACK PROBLEM – UNFEASIBLE

KNAPSACK PROBLEM: Given an integer-vector $X = (x_1, \ldots, x_n)$ and an integer c. Determine a binary vector $B = (b_1, \ldots, b_n)$ (if it exists) such that $XB^T = c$.

Knapsack problem with superincreasing vector – easy

Problem Given a superincreasing integer-vector $X = (x_1, \ldots, x_n)$ (i.e. $x_i > \sum_{j=1}^{i-1} x_j, i > 1$) and an integer c,

determine a binary vector $B = (b_1, \ldots, b_n)$ (if it exists) such that $XB^T = c$.

GENERAL KNAPSACK PROBLEM – UNFEASIBLE

KNAPSACK PROBLEM: Given an integer-vector $X = (x_1, \dots, x_n)$ and an integer c. Determine a binary vector $B = (b_1, \dots, b_n)$ (if it exists) such that $XB^T = c$.

Knapsack problem with superincreasing vector – easy

Problem Given a superincreasing integer-vector $X = (x_1, \dots, x_n)$ (i.e. $x_i > \sum_{j=1}^{i-1} x_j, i > 1$) and an integer c,

determine a binary vector $B = (b_1, \dots, b_n)$ (if it exists) such that $XB^T = c$.

Algorithm – to solve knapsack problems with superincreasing vectors:

for $i \leftarrow$ **downto 2 do**
    if $c \geq 2x_i$ **then** terminate $\{$no solution$\}$
        else if $c > x_i$ **then** $b_i \leftarrow 1; c \leftarrow c - x_i$;
            else $b_i = 0$;
if $c = x_1$ **then** $b_1 \leftarrow 1$
    else if $c = 0$ **then** $b_1 \leftarrow 0$;
        else terminate $\{$no solution$\}$

GENERAL KNAPSACK PROBLEM – UNFEASIBLE

KNAPSACK PROBLEM: Given an integer-vector $X = (x_1, \ldots, x_n)$ and an integer c. Determine a binary vector $B = (b_1, \ldots, b_n)$ (if it exists) such that $XB^T = c$.

Knapsack problem with superincreasing vector – easy

Problem Given a superincreasing integer-vector $X = (x_1, \ldots, x_n)$ (i.e. $x_i > \sum_{j=1}^{i-1} x_j, i > 1$) and an integer c,

determine a binary vector $B = (b_1, \ldots, b_n)$ (if it exists) such that $XB^T = c$.

Algorithm – to solve knapsack problems with superincreasing vectors:

for $i \leftarrow$ **downto 2 do**
    if $c \geq 2x_i$ **then** terminate $\{$no solution$\}$
        else if $c > x_i$ **then** $b_i \leftarrow 1; c \leftarrow c - x_i;$
            else $b_i = 0;$
if $c = x_1$ **then** $b_1 \leftarrow 1$
    else if $c = 0$ **then** $b_1 \leftarrow 0;$
        else terminate $\{$no solution$\}$

Example $X = (1,2,4,8,16,32,64,128,256,512)$ $c = 999$
$X = (1,3,5,10,20,41,94,199)$ $c = 242$

KNAPSACK ENCODING – BASIC IDEAS

Let a (knapsack) vector

$$A = (a_1, \ldots, a_n)$$

be given.

Encoding of a (binary) message $B = (b_1, b_2, \ldots, b_n)$ by A is done by the vector/vector multiplication:

$$AB^T = c$$

and results in the cryptotext c.

KNAPSACK ENCODING – BASIC IDEAS

Let a (knapsack) vector

$$A = (a_1, \dots, a_n)$$

be given.

Encoding of a (binary) message $B = (b_1, b_2, \dots, b_n)$ by A is done by the vector/vector multiplication:

$$AB^T = c$$

and results in the cryptotext c.

Decoding of c requires to solve the knapsack problem for the instant given by the knapsack vector A and the cryptotext c.

The problem is that decoding seems to be infeasible.

KNAPSACK ENCODING – BASIC IDEAS

Let a (knapsack) vector

$$A = (a_1, \ldots, a_n)$$

be given.

Encoding of a (binary) message $B = (b_1, b_2, \ldots, b_n)$ by A is done by the vector/vector multiplication:

$$AB^T = c$$

and results in the cryptotext c.

Decoding of c requires to solve the knapsack problem for the instant given by the knapsack vector A and the cryptotext c.

The problem is that decoding seems to be infeasible.

Example
If $A = (74, 82, 94, 83, 39, 99, 56, 49, 73, 99)$ and $B = (1100110101)$ then

$$AB^T =$$

DESIGN of KNAPSACK CRYPTOSYSTEMS

1. Choose a superincreasing vector $X = (x_1, \ldots, x_n)$.
2. Choose **m, u** such that $m > 2x_n,\ gcd(m, u) = 1$.
3. Compute u^{-1} mod $m, X' = (x'_1, \ldots, x'_n)$, $x'_i = \underbrace{\underbrace{ux_i}_{\text{diffusion}} \bmod m}_{\text{confusion}}$.

DESIGN of KNAPSACK CRYPTOSYSTEMS

1. Choose a superincreasing vector $X = (x_1, \dots, x_n)$.
2. Choose **m, u** such that $m > 2x_n$, $gcd(m, u) = 1$.
3. Compute u^{-1} mod $m, X' = (x'_1, \dots, x'_n)$, $x'_i = \underbrace{\underbrace{ux_i}_{\text{diffusion}} \bmod m}_{\text{confusion}}$.

Cryptosystem: X' – public key
X, u, m – trapdoor information

Encryption: of a binary vector w of length n: $c = X'w$

Decryption: compute $c' = u^{-1}c$ mod m
and solve the knapsack problem with X and c'.

1. Choose a superincreasing vector $X = (x_1, \ldots, x_n)$.
2. Choose **m, u** such that $m > 2x_n,\ gcd(m, u) = 1$.
3. Compute u^{-1} mod $m, X' = (x'_1, \ldots, x'_n)$, $x'_i = \underbrace{\underbrace{ux_i}_{\text{diffusion}} \bmod m}_{\text{confusion}}$.

Cryptosystem: X' – public key

X, u, m – trapdoor information

Encryption: of a binary vector w of length n: $c = X'w$

Decryption: compute $c' = u^{-1}c \bmod m$

and solve the knapsack problem with X and c'.

Lemma Let X, m, u, X', c, c' be as defined above. Then the knapsack problem instances (X, c') and (X', c) have at most one solution, and if one of them has a solution, then the second one has the same solution.

1. Choose a superincreasing vector $X = (x_1, \ldots, x_n)$.
2. Choose **m, u** such that $m > 2x_n$, $gcd(m, u) = 1$.
3. Compute u^{-1} mod m, $X' = (x'_1, \ldots, x'_n)$, $x'_i = \underbrace{\underbrace{ux_i}_{\text{diffusion}} \bmod m}_{\text{confusion}}$.

Cryptosystem: X' – public key

X, u, m – trapdoor information

Encryption: of a binary vector w of length n: $c = X'w$

Decryption: compute $c' = u^{-1}c \bmod m$

and solve the knapsack problem with X and c'.

Lemma Let X, m, u, X', c, c' be as defined above. Then the knapsack problem instances (X, c') and (X', c) have at most one solution, and if one of them has a solution, then the second one has the same solution.

Proof Let $X'w = c$. Then

$$c' \equiv u^{-1}c \equiv u^{-1}X'w \equiv u^{-1}uXw \equiv Xw (\text{mod } m).$$

Since X is superincreasing and $m > 2x_n$ we have

$$(Xw) \bmod m = Xw$$

and therefore

$$c' = Xw.$$

DESIGN of KNAPSACK CRYPTOSYSTEMS – EXAMPLE

Example $X = (1,2,4,9,18,35,75,151,302,606)$
$m = 1250$, $u = 41$
$X' = (41,82,164,369,738,185,575,1191,1132,1096)$

In order to encrypt an English plaintext, we first encode its letters by 5-bit numbers _ - 00000, A - 00001, B - 00010,. . . and then divide the resulting binary strings into blocks of length 10.

Plaintext: Encoding of AFRICA results in vectors

$$w_1 = (0000100110) \qquad w_2 = (1001001001) \qquad w_3 = (0001100001)$$

Encryption: $c_{1'} = X' w_1 = 3061 \qquad c_{2'} = X' w_2 = 2081 \qquad c_{3'} = X' w_3 = 2203$

Cryptotext: (3061,2081,2203)

Example $X = (1,2,4,9,18,35,75,151,302,606)$
$m = 1250$, $u = 41$
$X' = (41,82,164,369,738,185,575,1191,1132,1096)$

In order to encrypt an English plaintext, we first encode its letters by 5-bit numbers _ - 00000, A - 00001, B - 00010,... and then divide the resulting binary strings into blocks of length 10.

Plaintext: Encoding of AFRICA results in vectors

$$w_1 = (0000100110) \qquad w_2 = (1001001001) \qquad w_3 = (0001100001)$$

Encryption: $c_{1'} = X'w_1 = 3061 \qquad c_{2'} = X'w_2 = 2081 \qquad c_{3'} = X'w_3 = 2203$

Cryptotext: (3061,2081,2203)

Decryption of cryptotexts: (2163, 2116, 1870, 3599)

By multiplying with $u^{-1} = 61 \pmod{1250}$ we get new cryptotexts (several new c')

$$(693, 326, 320, 789)$$

And, in the binary form, solutions B of equations $XB^T = c'$ have the form

(1101001001, 0110100010, 0000100010, 1011100101)

Therefore, the resulting plaintext is:

ZIMBABWE

STORY of KNAPSACK

Invented: 1978 - Ralph C. Merkle, Martin Hellman
Patented: in 10 countries
Broken: 1982: Adi Shamir

New idea: iterated knapsack cryptosystem using hyper-reachable vectors.

Definition A knapsack vector $X' = (x_{1'}, \ldots, x_{n'})$ is obtained from a knapsack vector $X = (x_1, \ldots, x_n)$ by strong modular multiplication if

$$X_i' = ux_i \bmod m, i = 1, \ldots, n,$$

where

$$m > 2\sum_{i=1}^{n} x_i$$

and $gcd(u, m) = 1$. A knapsack vector X' is called hyper-reachable, if there is a sequence of knapsack vectors $X = x_0, x_1, \ldots, x_k = X'$,

where x_0 is a super-increasing vector and for $i = 1, \ldots, k$ x_i is obtained from x_{i-1} by a strong modular multiplication.

Iterated knapsack cryptosystem was broken in 1985 - E. Brickell

New ideas: dense knapsack cryptosystems. Density of a knapsack vector $X = (x_1, \ldots, x_n)$ is defined by $d(x) = \frac{n}{log(max\{x_i | 1 \le i \le n\})}$

Remark. Density of super-increasing vectors is $\le \frac{n}{n-1}$

KNAPSACK CRYPTOSYSTEM – COMMENTS

The term "knapsack" in the name of the cryptosystem is quite misleading.

By the Knapsack problem one usually understands the following problem:

Given n items with weights $w_1, w_2, \ldots, w_n$ and values $v_1, v_2, \ldots, v_n$ and a knapsack limit c, the task is to find a bit vector $(b_1, b_2, \ldots, b_n)$ such that $\sum_{i=1}^{n} b_i w_i \leq c$ and $\sum_{i=1}^{n} b_i v_i$ is as large as possible.

KNAPSACK CRYPTOSYSTEM – COMMENTS

The term “knapsack” in the name of the cryptosystem is quite misleading.

By the Knapsack problem one usually understands the following problem:

Given n items with weights $w_1, w_2, \ldots, w_n$ and values $v_1, v_2, \ldots, v_n$ and a knapsack limit c, the task is to find a bit vector $(b_1, b_2, \ldots, b_n)$ such that $\sum_{i=1}^{n} b_i w_i \leq c$ and $\sum_{i=1}^{n} b_i v_i$ is as large as possible.

The term subset problem is usually used for the problem used in our construction of the knapsack cryptosystem. It is well-known that the decision version of this problem is NP-complete.

The term “knapsack” in the name of the cryptosystem is quite misleading.

By the Knapsack problem one usually understands the following problem:

Given n items with weights $w_1, w_2, \ldots, w_n$ and values $v_1, v_2, \ldots, v_n$ and a knapsack limit c, the task is to find a bit vector $(b_1, b_2, \ldots, b_n)$ such that $\sum_{i=1}^{n} b_i w_i \leq c$ and $\sum_{i=1}^{n} b_i v_i$ is as large as possible.

The term subset problem is usually used for the problem used in our construction of the knapsack cryptosystem. It is well-known that the decision version of this problem is NP-complete.

Sometimes, for our main version of the knapsack problem the term Merkle-Hellman (Knapsack) Cryptosystem is used.

McEliece cryptosystem is based on a similar design principle as the Knapsack cryptosystem. McEliece cryptosystem is formed by transforming an easy to break cryptosystem into a cryptosystem that is hard to break because it seems to be based on a problem that is, in general, NP-hard.

The underlying fact is that the decision version of the decryption problem for linear codes is in general NP-complete. However, for special types of linear codes polynomial-time decryption algorithms exist. One such a class of linear codes, the so-called Goppa codes, are used to design McEliece cryptosystem.

Goppa codes are $[2^m, n - mt, 2t + 1]$-codes, where $n = 2^m$.
(McEliece suggested to use $m = 10, t = 50$.)

McELIECE CRYPTOSYSTEM – DESIGN

Goppa codes are $[2^m, n - mt, 2t + 1]$-codes, where $n = 2^m$.

Design of McEliece cryptosystems. Let

- G be a generating matrix for an $[n, k, d]$ Goppa code C;
- S be a $k \times k$ binary matrix invertible over Z_2;
- P be an $n \times n$ permutation matrix;
- $G' = SGP$.

Plaintexts: $P = (Z_2)^k$; cryptotexts: $C = (Z_2)^n$, key: $K = (G, S, P, G')$, message: w
G' is made public, G, S, P are kept secret.

Goppa codes are $[2^m, n - mt, 2t + 1]$-codes, where $n = 2^m$.

Design of McEliece cryptosystems. Let

- G be a generating matrix for an $[n, k, d]$ Goppa code C;
- S be a $k \times k$ binary matrix invertible over Z_2;
- P be an $n \times n$ permutation matrix;
- $G' = SGP$.

Plaintexts: $P = (Z_2)^k$; cryptotexts: $C = (Z_2)^n$, key: $K = (G, S, P, G')$, message: w
G' is made public, G, S, P are kept secret.

Encryption: $e_K(w, e) = wG' + e$, where e is any binary vector of length n & weight t.

Goppa codes are $[2^m, n - mt, 2t + 1]$-codes, where $n = 2^m$.

Design of McEliece cryptosystems. Let

- G be a generating matrix for an $[n, k, d]$ Goppa code C;
- S be a $k \times k$ binary matrix invertible over Z_2;
- P be an $n \times n$ permutation matrix;
- $G' = SGP$.

Plaintexts: $P = (Z_2)^k$; cryptotexts: $C = (Z_2)^n$, key: $K = (G, S, P, G')$, message: w
G' is made public, G, S, P are kept secret.

Encryption: $e_K(w, e) = wG' + e$, where e is any binary vector of length n & weight t.

Decryption of a cryptotext $c = wG' + e \in (Z_2)^n$.

1. Compute $c_1 = cP^{-1} = wSGPP^{-1} + eP^{-1} = wSG + eP^{-1}$
2. Decode c_1 to get $w_1 = wS$,
3. Compute $w = w_1 S^{-1}$

COMMENTS on McELIECE CRYPTOSYSTEM

1. Each irreducible polynomial over Z_2^m of degree t generates a Goppa code with distance at least $2t+1$.

COMMENTS on McELIECE CRYPTOSYSTEM

1. Each irreducible polynomial over Z_2^m of degree t generates a Goppa code with distance at least $2t+1$.
2. In the design of McEliece cryptosystem the goal of matrices S and C is to modify a generator matrix G for an easy-to-decode Goppa code to get a matrix that looks as a general random matrix for a linear code for which decoding problem is **NP**-complete.

COMMENTS on McELIECE CRYPTOSYSTEM

1. Each irreducible polynomial over Z_2^m of degree t generates a Goppa code with distance at least $2t+1$.
2. In the design of McEliece cryptosystem the goal of matrices S and C is to modify a generator matrix G for an easy-to-decode Goppa code to get a matrix that looks as a general random matrix for a linear code for which decoding problem is **NP**-complete.
3. An important novel and unique trick is an introduction, in the encoding process, of a random vector e that represents an introduction of up to t errors – such a number of errors that are correctable using the given Goppa code and this is the basic trick of the decoding process.

COMMENTS on McELIECE CRYPTOSYSTEM

1. Each irreducible polynomial over Z_2^m of degree t generates a Goppa code with distance at least $2t+1$.
2. In the design of McEliece cryptosystem the goal of matrices S and C is to modify a generator matrix G for an easy-to-decode Goppa code to get a matrix that looks as a general random matrix for a linear code for which decoding problem is **NP**-complete.
3. An important novel and unique trick is an introduction, in the encoding process, of a random vector e that represents an introduction of up to t errors – such a number of errors that are correctable using the given Goppa code and this is the basic trick of the decoding process.
4. Since P is a permutation matrix eP^{-1} has the same weight as e.

COMMENTS on McELIECE CRYPTOSYSTEM

1. Each irreducible polynomial over Z_2^m of degree t generates a Goppa code with distance at least $2t + 1$.
2. In the design of McEliece cryptosystem the goal of matrices S and C is to modify a generator matrix G for an easy-to-decode Goppa code to get a matrix that looks as a general random matrix for a linear code for which decoding problem is **NP**-complete.
3. An important novel and unique trick is an introduction, in the encoding process, of a random vector e that represents an introduction of up to t errors – such a number of errors that are correctable using the given Goppa code and this is the basic trick of the decoding process.
4. Since P is a permutation matrix eP^{-1} has the same weight as e.
5. As already mentioned, McEliece suggested to use a Goppa code with $m = 10$ and $t = 50$. This provides a $[1024, 524, 101]$-code. Each plaintext is then a 524-bit string, each cryptotext is a 1024-bit string. The public key is an 524×1024 matrix.

COMMENTS on McELIECE CRYPTOSYSTEM

1. Each irreducible polynomial over Z_2^m of degree t generates a Goppa code with distance at least $2t+1$.
2. In the design of McEliece cryptosystem the goal of matrices S and C is to modify a generator matrix G for an easy-to-decode Goppa code to get a matrix that looks as a general random matrix for a linear code for which decoding problem is **NP**-complete.
3. An important novel and unique trick is an introduction, in the encoding process, of a random vector e that represents an introduction of up to t errors – such a number of errors that are correctable using the given Goppa code and this is the basic trick of the decoding process.
4. Since P is a permutation matrix eP^{-1} has the same weight as e.
5. As already mentioned, McEliece suggested to use a Goppa code with $m = 10$ and $t = 50$. This provides a $[1024, 524, 101]$-code. Each plaintext is then a 524-bit string, each cryptotext is a 1024-bit string. The public key is an 524×1024 matrix.
6. Observe that the number of potential matrices S and P is so large that probability of guessing these matrices is smaller that probability of guessing correct plaintext!!!

COMMENTS on McELIECE CRYPTOSYSTEM

1. Each irreducible polynomial over Z_2^m of degree t generates a Goppa code with distance at least $2t+1$.
2. In the design of McEliece cryptosystem the goal of matrices S and C is to modify a generator matrix G for an easy-to-decode Goppa code to get a matrix that looks as a general random matrix for a linear code for which decoding problem is **NP**-complete.
3. An important novel and unique trick is an introduction, in the encoding process, of a random vector e that represents an introduction of up to t errors – such a number of errors that are correctable using the given Goppa code and this is the basic trick of the decoding process.
4. Since P is a permutation matrix eP^{-1} has the same weight as e.
5. As already mentioned, McEliece suggested to use a Goppa code with $m = 10$ and $t = 50$. This provides a $[1024, 524, 101]$-code. Each plaintext is then a 524-bit string, each cryptotext is a 1024-bit string. The public key is an 524×1024 matrix.
6. Observe that the number of potential matrices S and P is so large that probability of guessing these matrices is smaller that probability of guessing correct plaintext!!!
7. It can be shown that it is not safe to encrypt twice the same plaintext with the same public key (and different error vectors).

1. Public-key cryptosystems can never provide unconditional security. This is because an eavesdropper, on observing a cryptotext c can encrypt each possible plaintext by the encryption algorithm e_A until he finds c such that $e_A(w) = c$.

FINAL COMMENTS

1. Public-key cryptosystems can never provide unconditional security. This is because an eavesdropper, on observing a cryptotext c can encrypt each possible plaintext by the encryption algorithm e_A until he finds c such that $e_A(w) = c$.
2. One-way functions exist if and only if $\mathbf{P} = \mathbf{UP}$, where UP is the class of languages accepted by **unambiguous polynomial time bounded nondeterministic Turing machine.**

FINAL COMMENTS

1. Public-key cryptosystems can never provide unconditional security. This is because an eavesdropper, on observing a cryptotext c can encrypt each possible plaintext by the encryption algorithm e_A until he finds c such that $e_A(w) = c$.
2. One-way functions exist if and only if $\mathbf{P} = \mathbf{UP}$, where UP is the class of languages accepted by **unambiguous polynomial time bounded nondeterministic Turing machine.**
3. There are actually two types of keys in practical use: A session key is used for sending a particular message (or few of them). A master key is usually used to generate several session keys.

FINAL COMMENTS

1. Public-key cryptosystems can never provide unconditional security. This is because an eavesdropper, on observing a cryptotext c can encrypt each possible plaintext by the encryption algorithm e_A until he finds c such that $e_A(w) = c$.
2. One-way functions exist if and only if $\mathbf{P} = \mathbf{UP}$, where UP is the class of languages accepted by **unambiguous polynomial time bounded nondeterministic Turing machine.**
3. There are actually two types of keys in practical use: A session key is used for sending a particular message (or few of them). A master key is usually used to generate several session keys.
4. Session keys are usually generated when actually required and discarded after their use. Session keys are usually keys of a secret-key cryptosystem.

FINAL COMMENTS

1. Public-key cryptosystems can never provide unconditional security. This is because an eavesdropper, on observing a cryptotext c can encrypt each possible plaintext by the encryption algorithm e_A until he finds c such that $e_A(w) = c$.
2. One-way functions exist if and only if $\mathbf{P} = \mathbf{UP}$, where UP is the class of languages accepted by **unambiguous polynomial time bounded nondeterministic Turing machine.**
3. There are actually two types of keys in practical use: A session key is used for sending a particular message (or few of them). A master key is usually used to generate several session keys.
4. Session keys are usually generated when actually required and discarded after their use. Session keys are usually keys of a secret-key cryptosystem.
5. Master keys are usually used for longer time and need therefore be carefully stored. Master keys are usually keys of a public-key cryptosystem.

SATELLITE VERSION of ONE-TIME PAD

Suppose a satellite produces and broadcasts several random sequences of bits at a rate fast enough that no computer can store more than a small fraction of the output.

If Alice wants to send a message to Bob they first agree, using a public key cryptography, on a method of sampling bits from the satellite outputs.

Alice and Bob use this method to generate a random key and they use it with ONE-TIME PAD for encryption.

By the time Eve decrypted their public key communications, random streams produced by the satellite and used by Alice and Bob to get the secret key have disappeared, and therefore there is no way for Eve to make decryption.

The point is that satellites produce so large amount of date that Eve cannot store all of them

RSA CRYPTOSYSTEM

The most important public-key cryptosystem is the RSA cryptosystem on which one can also illustrate a variety of important ideas of modern public-key cryptography.

For example, we will discuss various possible attacks on the RSA cryptosystem and problems related to security of RSA.

A special attention will be given in Chapter 7 to the problem of factorization of integers that play such an important role for security of RSA.

In doing that we will illustrate modern distributed techniques to factorize very large integers.

Invented in 1978 by Rivest, Shamir, Adleman

Basic idea: prime multiplication is very easy, integer factorization seems to be unfeasible.

DESIGN and USE of RSA CRYPTOSYSTEM

Invented in 1978 by Rivest, Shamir, Adleman

Basic idea: prime multiplication is very easy, integer factorization seems to be unfeasible.

Design of RSA cryptosystems

1. Choose two large s-bit primes p,q, s in [512,1024], and denote

$$n = pq, \phi(n) = (p-1)(q-1)$$

2. Choose a large d such that

$$\gcd(d, \phi(n)) = 1$$

and compute

$$e = d^{-1} (\bmod\ \phi(n))$$

Public key: n (modulus), e (encryption exponent)
Trapdoor information: p, q, d (decryption exponent)

DESIGN and USE of RSA CRYPTOSYSTEM

Invented in 1978 by Rivest, Shamir, Adleman
Basic idea: prime multiplication is very easy, integer factorization seems to be unfeasible.

Design of RSA cryptosystems

1. Choose two large s-bit primes p,q, s in [512,1024], and denote
$$n = pq, \phi(n) = (p-1)(q-1)$$
2. Choose a large d such that
$$\gcd(d, \phi(n)) = 1$$

and compute
$$e = d^{-1} (\text{mod } \phi(n))$$

Public key: n (modulus), e (encryption exponent)
Trapdoor information: p, q, d (decryption exponent)

Plaintext w
Encryption: cryptotext $c = w^e \bmod n$
Decryption: plaintext $w = c^d \bmod n$

DESIGN and USE of RSA CRYPTOSYSTEM

Invented in 1978 by Rivest, Shamir, Adleman

Basic idea: prime multiplication is very easy, integer factorization seems to be unfeasible.

Design of RSA cryptosystems

1. Choose two large s-bit primes p,q, s in [512,1024], and denote

$$n = pq, \phi(n) = (p-1)(q-1)$$

2. Choose a large d such that

$$\gcd(d, \phi(n)) = 1$$

and compute

$$e = d^{-1} (\mathrm{mod}\ \phi(n))$$

Public key: n (modulus), e (encryption exponent)
Trapdoor information: p, q, d (decryption exponent)

Plaintext w
Encryption: cryptotext $c = w^e \bmod n$
Decryption: plaintext $w = c^d \bmod n$

Details: A plaintext is first encoded as a word over the alphabet $\{0, 1, \ldots, 9\}$, then divided into blocks of length $i-1$, where $10^{i-1} < n < 10^i$. Each block is taken as an integer and decrypted using modular exponentiation.

CORRECTNESS of RSA

Let $c = w^e \bmod n$ be the cryptotext for a plaintext w, in the cryptosystem with

$$n = pq, ed \equiv 1 \pmod{\phi(n)}, \gcd(d, \phi(n)) = 1$$

In such a case

$$w \equiv c^d \bmod n$$

and, if the decryption is unique, $w = c^d \bmod n$.

Let $c = w^e \bmod n$ be the cryptotext for a plaintext w, in the cryptosystem with

$$n = pq, ed \equiv 1 \pmod{\phi(n)}, \gcd(d, \phi(n)) = 1$$

In such a case

$$w \equiv c^d \bmod n$$

and, if the decryption is unique, $w = c^d \bmod n$.

Proof Since $ed \equiv 1 \pmod{\phi(n)}$, there exist a $j \in N$ such that $ed = j\phi(n) + 1$.

- Case 1. Neither p nor q divides w.
  In such a case $\gcd(n, w) = 1$ and by the Euler's Totient Theorem we get that

$$c^d = w^{ed} = w^{j\phi(n)+1} \equiv w \pmod{n}$$

Let $c = w^e \bmod n$ be the cryptotext for a plaintext w, in the cryptosystem with

$$n = pq, ed \equiv 1 \pmod{\phi(n)}, \gcd(d, \phi(n)) = 1$$

In such a case

$$w \equiv c^d \bmod n$$

and, if the decryption is unique, $w = c^d \bmod n$.

Proof Since $ed \equiv 1 \pmod{\phi(n)}$, there exist a $j \in N$ such that $ed = j\phi(n) + 1$.

- Case 1. Neither p nor q divides w.
  In such a case $\gcd(n, w) = 1$ and by the Euler's Totient Theorem we get that

$$c^d = w^{ed} = w^{j\phi(n)+1} \equiv w \pmod{n}$$

- Case 2. Exactly one of p, q divides w – say p.
  In such a case $w^{ed} \equiv w \pmod{p}$ and by Fermat's Little theorem $w^{q-1} \equiv 1 \pmod{q}$

$$\begin{aligned} \Rightarrow w^{q-1} \equiv 1 \pmod{q} &\Rightarrow w^{\phi(n)} \equiv 1 \pmod{q} \\ &\Rightarrow w^{j\phi(n)} \equiv 1 \pmod{q} \\ &\Rightarrow w^{ed} \equiv w \pmod{q} \end{aligned}$$

  Therefore: $w \equiv w^{ed} \equiv c^d \pmod{n}$

CORRECTNESS of RSA

Let $c = w^e \text{mod } n$ be the cryptotext for a plaintext w, in the cryptosystem with

$$n = pq, ed \equiv 1 \pmod{\phi(n)}, \gcd(d, \phi(n)) = 1$$

In such a case

$$w \equiv c^d \text{ mod } n$$

and, if the decryption is unique, $w = c^d \text{mod } n$.

Proof Since $ed \equiv 1 \pmod{\phi(n)}$, there exist a $j \in N$ such that $ed = j\phi(n) + 1$.

- Case 1. Neither p nor q divides w.
  In such a case $\gcd(n, w) = 1$ and by the Euler's Totient Theorem we get that

$$c^d = w^{ed} = w^{j\phi(n)+1} \equiv w \pmod{n}$$

- Case 2. Exactly one of p, q divides w – say p.
  In such a case $w^{ed} \equiv w \pmod{p}$ and by Fermat's Little theorem $w^{q-1} \equiv 1 \pmod{q}$

$$\begin{aligned} \Rightarrow w^{q-1} \equiv 1 \pmod{q} \Rightarrow w^{\phi(n)} &\equiv 1 \pmod{q} \\ \Rightarrow w^{j\phi(n)} &\equiv 1 \pmod{q} \\ \Rightarrow w^{ed} &\equiv w \pmod{q} \end{aligned}$$

  Therefore: $w \equiv w^{ed} \equiv c^d \pmod{n}$
- Case 3. Both p, q divide w.
  This cannot happen because, by our assumption, $w < n$.

DESIGN and USE of RSA CRYPTOSYSTEM

Example of the design and of the use of RSA cryptosystems.

- By choosing $p = 41, q = 61$ we get $n = 2501, \phi(n) = 2400$
- By choosing $d = 2087$ we get $e = 23$
- By choosing $d = 2069$ we get $e = 29$
- By choosing other values of d we would get other values of e.

Example of the design and of the use of RSA cryptosystems.

- By choosing $p = 41, q = 61$ we get $n = 2501, \phi(n) = 2400$
- By choosing $d = 2087$ we get $e = 23$
- By choosing $d = 2069$ we get $e = 29$
- By choosing other values of d we would get other values of e.

Let us choose the first pair of encryption/decryption exponents ($e = 23$ and $d = 2087$).

Plaintext: KARLSRUHE   Encoding: 100017111817200704

Since $10^3 < n < 10^4$, the numerical plaintext is divided into blocks of 3 digits $\Rightarrow$ 6 plaintext integers are obtained

$$100, 017, 111, 817, 200, 704$$

Example of the design and of the use of RSA cryptosystems.

- By choosing $p = 41, q = 61$ we get $n = 2501, \phi(n) = 2400$
- By choosing $d = 2087$ we get $e = 23$
- By choosing $d = 2069$ we get $e = 29$
- By choosing other values of d we would get other values of e.

Let us choose the first pair of encryption/decryption exponents ($e = 23$ and $d = 2087$).

Plaintext: KARLSRUHE Encoding: 100017111817200704

Since $10^3 < n < 10^4$, the numerical plaintext is divided into blocks of 3 digits $\Rightarrow$ 6 plaintext integers are obtained

$$100, 017, 111, 817, 200, 704$$

Encryption:

$$100^{23} \bmod 2501, 17^{23} \bmod 2501, 111^{23} \bmod 2501$$
$$817^{23} \bmod 2501, 200^{23} \bmod 2501, 704^{23} \bmod 2501$$

provides cryptotexts:

$$2306, 1893, 621, 1380, 490, 313$$

Example of the design and of the use of RSA cryptosystems.

- By choosing $p = 41, q = 61$ we get $n = 2501, \phi(n) = 2400$
- By choosing $d = 2087$ we get $e = 23$
- By choosing $d = 2069$ we get $e = 29$
- By choosing other values of d we would get other values of e.

Let us choose the first pair of encryption/decryption exponents ($e = 23$ and $d = 2087$).

Plaintext: KARLSRUHE Encoding: 100017111817200704

Since $10^3 < n < 10^4$, the numerical plaintext is divided into blocks of 3 digits $\Rightarrow$ 6 plaintext integers are obtained

$$100, 017, 111, 817, 200, 704$$

Encryption:

$$100^{23} \bmod 2501, 17^{23} \bmod 2501, 111^{23} \bmod 2501$$
$$817^{23} \bmod 2501, 200^{23} \bmod 2501, 704^{23} \bmod 2501$$

provides cryptotexts:

$$2306, 1893, 621, 1380, 490, 313$$

Decryption:

$$2306^{2087} \bmod 2501 = 100, 1893^{2087} \bmod 2501 = 17$$
$$621^{2087} \bmod 2501 = 111, 1380^{2087} \bmod 2501 = 817$$
$$490^{2087} \bmod 2501 = 200, 313^{2087} \bmod 2501 = 704$$

One of the first descriptions of RSA was in the paper.

Martin Gardner: Mathematical games, Scientific American, 1977

and in this paper RSA inventors presented the following challenge.

Decrypt the cryptotext:

9686 9613 7546 2206 1477 1409 2225 4355 8829 0575 9991 1245 7431 9874 6951 2093
0816 2982 2514 5708 3569 3147 6622 8839 8962 8013 3919 9055 1829 9451 5781 5154

RSA CHALLENGE

One of the first descriptions of RSA was in the paper.

Martin Gardner: Mathematical games, Scientific American, 1977

and in this paper RSA inventors presented the following challenge.

Decrypt the cryptotext:

9686 9613 7546 2206 1477 1409 2225 4355 8829 0575 9991 1245 7431 9874 6951 2093 0816 2982 2514 5708 3569 3147 6622 8839 8962 8013 3919 9055 1829 9451 5781 5154

encrypted using the RSA cryptosystem with 129 digit number, called also RSA129

n: 114 381 625 757 888 867 669 235 779 976 146 612 010 218 296 721 242 362 562 561 842 935 706 935 245 733 897 830 597 123 513 958 705 058 989 075 147 599 290 026 879 543 541.

and with $e = 9007$.

The problem was solved in 1994 by first factorizing n into one 64-bit prime and one 65-bit prime, and then computing the plaintext

THE MAGIC WORDS ARE SQUEMISH OSSIFRAGE

HOW to DESIGN REALLY GOOD RSA CRYPTOSYSTEMS?

1. How to choose large primes p, q?
   Choose randomly a large integer p, and verify, using a randomized algorithm, whether p is prime. If not, check $p+2, p+4, \ldots$ From the Prime Number Theorem it follows that there are approximately

$$\frac{2^d}{\log 2^d} - \frac{2^{d-1}}{\log 2^{d-1}}$$

   d bit primes. (A probability that a 512-bit number is prime is 0.00562.)

HOW to DESIGN REALLY GOOD RSA CRYPTOSYSTEMS?

1. How to choose large primes p, q?
   Choose randomly a large integer p, and verify, using a randomized algorithm, whether p is prime. If not, check $p+2, p+4, \ldots$ From the Prime Number Theorem it follows that there are approximately

$$\frac{2^d}{\log 2^d} - \frac{2^{d-1}}{\log 2^{d-1}}$$

   d bit primes. (A probability that a 512-bit number is prime is 0.00562.)
2. What kind of relations should be between p and q?
   - 2.1 Difference $|p-q|$ should be neither too small nor too large.
   - 2.2 $\gcd(p-1, q-1)$ should not be large.
   - 2.3 Both $p-1$ and $q-1$ should contain large prime factors.
   - 2.4 Quite ideal case: q, p should be safe primes - such that also $(p–1)/2$ and $(q-1)/2$ are primes. ($83, 107, 10^{100} - 166517$ are examples of safe primes).

HOW to DESIGN REALLY GOOD RSA CRYPTOSYSTEMS?

1. How to choose large primes p, q?
   Choose randomly a large integer p, and verify, using a randomized algorithm, whether p is prime. If not, check $p+2, p+4, \ldots$ From the Prime Number Theorem it follows that there are approximately

$$\frac{2^d}{\log 2^d} - \frac{2^{d-1}}{\log 2^{d-1}}$$

   d bit primes. (A probability that a 512-bit number is prime is 0.00562.)
2. What kind of relations should be between p and q?
   - 2.1 Difference $|p-q|$ should be neither too small nor too large.
   - 2.2 $\gcd(p-1, q-1)$ should not be large.
   - 2.3 Both $p-1$ and $q-1$ should contain large prime factors.
   - 2.4 Quite ideal case: q, p should be safe primes - such that also $(p–1)/2$ and $(q-1)/2$ are primes. ($83, 107, 10^{100} - 166517$ are examples of safe primes).
3. How to choose e and d?
   - 3.1 Neither d nor e should be small.
   - 3.2 d should not be smaller than $n^{\frac{1}{4}}$. (For $d < n^{\frac{1}{4}}$ a polynomial time algorithm is known to determine d).

PRIME RECOGNITION and FACTORIZATION

The key problems for the development of RSA cryptosystem are that of prime recognition and integer factorization.

On August 2002, the first polynomial time algorithm was discovered that allows to determine whether a given m bit integer is a prime. Algorithm works in time $O(m^{12})$.

Fast randomized algorithms for prime recognition has been known since 1977. One of the simplest one is due to Rabin and will be presented later.

PRIME RECOGNITION and FACTORIZATION

The key problems for the development of RSA cryptosystem are that of prime recognition and integer factorization.

On August 2002, the first polynomial time algorithm was discovered that allows to determine whether a given m bit integer is a prime. Algorithm works in time $O(m^{12})$.

Fast randomized algorithms for prime recognition has been known since 1977. One of the simplest one is due to Rabin and will be presented later.

For integer factorization situation is somehow different.

- No polynomial time classical algorithm is known.
- Simple, but not efficient factorization algorithms are known.
- Several sophisticated distributed factorization algorithms are known that allowed to factorize, using enormous computation power, surprisingly large integers.
- Progress in integer factorization, due to progress in algorithms and technology, has been recently enormous.
- Polynomial time quantum algorithms for integer factorization are known since 1994 (P. Shor).

Several simple and some sophisticated factorization algorithms will be presented and illustrated in the following.

RABIN-MILLER's PRIME RECOGNITION

Rabin-Miller's Monte Carlo prime recognition algorithm is based on the following result from the number theory.

Lemma Let $n \in N$. Denote, for $1 \leq x \leq n$, by $C(x)$ the condition:
Either $x^{n-1} \neq 1 \pmod{n}$, or there is an $m = \frac{n-1}{2^i}$ for some i, such that $\gcd(n, x^m - 1) \neq 1$
If $C(x)$ holds for some $1 \leq x \leq n$, then n is not a prime. If n is not a prime, then $C(x)$ holds for at least half of x between 1 and n.

Rabin-Miller's Monte Carlo prime recognition algorithm is based on the following result from the number theory.

Lemma Let $n \in N$. Denote, for $1 \leq x \leq n$, by $C(x)$ the condition:
Either $x^{n-1} \neq 1 \pmod{n}$, or there is an $m = \frac{n-1}{2^i}$ for some i, such that $\gcd(n, x^m - 1) \neq 1$
If $C(x)$ holds for some $1 \leq x \leq n$, then n is not a prime. If n is not a prime, then $C(x)$ holds for at least half of x between 1 and n.

Algorithm:

Choose randomly integers $x_1, x_2, \ldots, x_m$ such that $1 \leq x_i \leq n$.
For each x_i determine whether $C(x_i)$ holds.

Rabin-Miller's Monte Carlo prime recognition algorithm is based on the following result from the number theory.

Lemma Let $n \in N$. Denote, for $1 \leq x \leq n$, by $C(x)$ the condition:
Either $x^{n-1} \neq 1 \pmod{n}$, or there is an $m = \frac{n-1}{2^i}$ for some i, such that $\gcd(n, x^m - 1) \neq 1$
If $C(x)$ holds for some $1 \leq x \leq n$, then n is not a prime. If n is not a prime, then $C(x)$ holds for at least half of x between 1 and n.

Algorithm:

Choose randomly integers $x_1, x_2, \ldots, x_m$ such that $1 \leq x_i \leq n$.
For each x_i determine whether $C(x_i)$ holds.

Claim: If $C(x_i)$ holds for some i, then n is not a prime for sure. Otherwise n is declared to be prime. Probability that this is not the case is 2^{-m}.

FACTORIZATION of 512-BITS and 663-BITS NUMBERS

On August 22, 1999, a team of scientists from 6 countries found, after 7 months of computing, using 300 very fast SGI and SUN workstations and Pentium II, factors of the so-called RSA-155 number with 512 bits (about 155 digits).

FACTORIZATION of 512-BITS and 663-BITS NUMBERS

On August 22, 1999, a team of scientists from 6 countries found, after 7 months of computing, using 300 very fast SGI and SUN workstations and Pentium II, factors of the so-called RSA-155 number with 512 bits (about 155 digits).

RSA-155 was a number from a Challenge list issue by the US company RSA Data Security and "represented" 95% of 512-bit numbers used as the key to protect electronic commerce and financial transmissions on Internet.

FACTORIZATION of 512-BITS and 663-BITS NUMBERS

On August 22, 1999, a team of scientists from 6 countries found, after 7 months of computing, using 300 very fast SGI and SUN workstations and Pentium II, factors of the so-called RSA-155 number with 512 bits (about 155 digits).

RSA-155 was a number from a Challenge list issue by the US company RSA Data Security and "represented" 95% of 512-bit numbers used as the key to protect electronic commerce and financial transmissions on Internet.

Factorization of RSA-155 would require in total 37 years of computing time on a single computer.

FACTORIZATION of 512-BITS and 663-BITS NUMBERS

On August 22, 1999, a team of scientists from 6 countries found, after 7 months of computing, using 300 very fast SGI and SUN workstations and Pentium II, factors of the so-called RSA-155 number with 512 bits (about 155 digits).

RSA-155 was a number from a Challenge list issue by the US company RSA Data Security and "represented" 95% of 512-bit numbers used as the key to protect electronic commerce and financial transmissions on Internet.

Factorization of RSA-155 would require in total 37 years of computing time on a single computer.

When in 1977 Rivest and his colleagues challenged the world to factor RSA-129, they estimated that, using knowledge of that time, factorization of RSA-129 would require 10^{16} years.

FACTORIZATION of 512-BITS and 663-BITS NUMBERS

On August 22, 1999, a team of scientists from 6 countries found, after 7 months of computing, using 300 very fast SGI and SUN workstations and Pentium II, factors of the so-called RSA-155 number with 512 bits (about 155 digits).

RSA-155 was a number from a Challenge list issue by the US company RSA Data Security and "represented" 95% of 512-bit numbers used as the key to protect electronic commerce and financial transmissions on Internet.

Factorization of RSA-155 would require in total 37 years of computing time on a single computer.

When in 1977 Rivest and his colleagues challenged the world to factor RSA-129, they estimated that, using knowledge of that time, factorization of RSA-129 would require 10^{16} years.

In 2005 RSA-200, a 663-bits number, was factorized by a team of German Federal Agency for Information Technology Security, using CPU of 80 AMD Opterons.

LARGE NUMBERS

Hindus named many large numbers - one having 153 digits.

Romans initially had no terms for numbers larger than 10^4.

Greeks had a popular belief that no number is larger than the total count of sand grains needed to fill the universe.

Large numbers with special names:

duotrigintillion=googol-10^{100} googolplex$-10^{10^{100}}$

LARGE NUMBERS

Hindus named many large numbers - one having 153 digits.

Romans initially had no terms for numbers larger than 10^4.

Greeks had a popular belief that no number is larger than the total count of sand grains needed to fill the universe.

Large numbers with special names:

duotrigintillion=googol-10^{100} googolplex$-10^{10^{100}}$

FACTORIZATION of very large NUMBERS

W. Keller factorized F_{23471} which has 10^{7000} digits.

J. Harley factorized: $10^{10^{1000}} + 1$.

One factor: $316,912,650,057,350,374,175,801,344,000,001$

1992 E. Crandal, Doenias proved, using a computer that F_{22}, which has more than million of digits, is composite (but no factor of F_{22} is known).

Number $10^{10^{10^{34}}}$ was used to develop a theory of the distribution of prime numbers.

Claim 1. Difference $|p-q|$ should not be small.

Indeed, if $|p-q|$ is small, and $p>q$, then $\frac{(p+q)}{2}$ is only slightly larger than $\sqrt{n}$ because

$$\frac{(p+q)^2}{4}-n=\frac{(p-q)^2}{4}$$

In addition $\frac{(p+q)^2}{4}-n$ is a square, say y^2.
In order to factor n, it is then enough to test $x>\sqrt{n}$ until x is found such that x^2-n is a square, say y^2. In such a case

$$p+q=2x, p-q=2y \qquad \text{and therefore } p=x+y, q=x-y.$$

DESIGN OF GOOD RSA CRYPTOSYSTEMS

Claim 1. Difference $|p - q|$ should not be small.

Indeed, if $|p - q|$ is small, and $p > q$, then $\frac{(p+q)}{2}$ is only slightly larger than $\sqrt{n}$ because

$$\frac{(p+q)^2}{4} - n = \frac{(p-q)^2}{4}$$

In addition $\frac{(p+q)^2}{4} - n$ is a square, say y^2.
In order to factor n, it is then enough to test $x > \sqrt{n}$ until x is found such that $x^2 - n$ is a square, say y^2. In such a case

$$p + q = 2x, p - q = 2y \qquad \text{and therefore } p = x + y, q = x - y.$$

Claim 2. $\gcd(p-1, q-1)$ should not be large.

Indeed, in the opposite case $s = \operatorname{lcm}(p-1, q-1)$ is much smaller than $\phi(n)$ If

$$d'e \equiv 1 \bmod s,$$

then, for some integer k,

$$c^d \equiv w^{ed} \equiv w^{ks+1} \equiv w \bmod n$$

since $p - 1|s, q - 1|s$ and therefore $w^{ks} \equiv 1 \bmod p$ and $w^{ks+1} \equiv w \bmod q$. Hence, d' can serve as a decryption exponent.
Moreover, in such a case s can be obtained by testing.

DESIGN OF GOOD RSA CRYPTOSYSTEMS

Claim 1. Difference $|p-q|$ should not be small.

Indeed, if $|p-q|$ is small, and $p>q$, then $\frac{(p+q)}{2}$ is only slightly larger than $\sqrt{n}$ because

$$\frac{(p+q)^2}{4}-n=\frac{(p-q)^2}{4}$$

In addition $\frac{(p+q)^2}{4}-n$ is a square, say y^2.
In order to factor n, it is then enough to test $x>\sqrt{n}$ until x is found such that x^2-n is a square, say y^2. In such a case

$$p+q=2x, p-q=2y \qquad \text{and therefore } p=x+y, q=x-y.$$

Claim 2. $\gcd(p-1,q-1)$ should not be large.

Indeed, in the opposite case $s=\operatorname{lcm}(p-1,q-1)$ is much smaller than $\phi(n)$ If

$$d'e\equiv 1 \bmod s,$$

then, for some integer k,

$$c^d\equiv w^{ed}\equiv w^{ks+1}\equiv w \bmod n$$

since $p-1|s, q-1|s$ and therefore $w^{ks}\equiv 1 \bmod p$ and $w^{ks+1}\equiv w \bmod q$. Hence, d' can serve as a decryption exponent.
Moreover, in such a case s can be obtained by testing.
Question Is there enough primes (to choose again and again new ones)?
No problem, the number of primes of length 512 bit or less exceeds 10^{150}.

HOW IMPORTANT is FACTORIZATION for BREAKING RSA?

1. If integer factorization is feasible, then RSA is breakable.

HOW IMPORTANT is FACTORIZATION for BREAKING RSA?

1. If integer factorization is feasible, then RSA is breakable.
2. There is no proof that factorization is indeed needed to break RSA.

HOW IMPORTANT is FACTORIZATION for BREAKING RSA?

1. If integer factorization is feasible, then RSA is breakable.
2. There is no proof that factorization is indeed needed to break RSA.
3. If a method of breaking RSA would provide an effective way to get a trapdoor information, then factorization could be done effectively.

Theorem Any algorithm to compute $\phi(n)$ can be used to factor integers with the same complexity.

Theorem Any algorithm for computing d can be converted into a break randomized algorithm for factoring integers with the same complexity.

HOW IMPORTANT is FACTORIZATION for BREAKING RSA?

1. If integer factorization is feasible, then RSA is breakable.
2. There is no proof that factorization is indeed needed to break RSA.
3. If a method of breaking RSA would provide an effective way to get a trapdoor information, then factorization could be done effectively.

   Theorem Any algorithm to compute $\phi(n)$ can be used to factor integers with the same complexity.

   Theorem Any algorithm for computing d can be converted into a break randomized algorithm for factoring integers with the same complexity.
4. There are setups in which RSA can be broken without factoring modulus n.

   Example An agency chooses p, q and computes a modulus $n = pq$ that is publicized and common to all users $U_1, U_2, \ldots$ and also encryption exponents $e_1, e_2, \ldots$ are publicized. Each user U_i gets his decryption exponent d_i.

   In such a setting any user is able to find in deterministic quadratic time another user's decryption exponent.

None of the numerous attempts to develop attacks on RSA has turned out to be successful.

There are various results showing that it is impossible to obtain even only partial information about the plaintext from the cryptotext produced by the RSA cryptosystem.

We will show that were the following two functions, that are computationally polynomially equivalent, be efficiently computable, then the RSA cryptosystem with the encryption (decryption) exponents $e_k(d_k)$ would be breakable.

$$parity_{ek}(c) = \text{the least significant bit of such an } w \text{ that } e_k(w) = c;$$
$$half_{ek}(c) = 0 \text{ if } 0 \leq w < \frac{n}{2} \text{ and } half_{ek}(c) = 1 \text{ if } \frac{n}{2} \leq w \leq n-1$$

SECURITY of RSA in PRACTICE

None of the numerous attempts to develop attacks on RSA has turned out to be successful.

There are various results showing that it is impossible to obtain even only partial information about the plaintext from the cryptotext produced by the RSA cryptosystem.

We will show that were the following two functions, that are computationally polynomially equivalent, be efficiently computable, then the RSA cryptosystem with the encryption (decryption) exponents $e_k(d_k)$ would be breakable.

$$parity_{ek}(c) = \text{the least significant bit of such an } w \text{ that } e_k(w) = c;$$
$$half_{ek}(c) = 0 \text{ if } 0 \leq w < \tfrac{n}{2} \text{ and } half_{ek}(c) = 1 \text{ if } \tfrac{n}{2} \leq w \leq n-1$$

We show two important properties of the functions *half* and *parity*.

1. Polynomial time computational equivalence of the functions half and parity follows from the following identities

$$half_{ek}(c) = parity_{ek}((c \times e_k(2)) \bmod n$$

$$parity_{ek}(c) = half_{ek}((c \times e_k(\frac{1}{2})) \bmod n$$

and the multiplicative rule $e_k(w_1)e_k(w_2) = e_k(w_1 w_2)$.

SECURITY of RSA in PRACTICE

None of the numerous attempts to develop attacks on RSA has turned out to be successful.

There are various results showing that it is impossible to obtain even only partial information about the plaintext from the cryptotext produced by the RSA cryptosystem.

We will show that were the following two functions, that are computationally polynomially equivalent, be efficiently computable, then the RSA cryptosystem with the encryption (decryption) exponents $e_k(d_k)$ would be breakable.

$$parity_{ek}(c) = \text{the least significant bit of such an } w \text{ that } e_k(w) = c;$$
$$half_{ek}(c) = 0 \text{ if } 0 \leq w < \tfrac{n}{2} \text{ and } half_{ek}(c) = 1 \text{ if } \tfrac{n}{2} \leq w \leq n-1$$

We show two important properties of the functions *half* and *parity*.

1. Polynomial time computational equivalence of the functions half and parity follows from the following identities

$$half_{ek}(c) = parity_{ek}((c \times e_k(2)) \bmod n$$

$$parity_{ek}(c) = half_{ek}((c \times e_k(\frac{1}{2})) \bmod n$$

and the multiplicative rule $e_k(w_1)e_k(w_2) = e_k(w_1 w_2)$.

2. There is an efficient algorithm to determine plaintexts w from the cryptotexts c obtained by RSA-decryption provided efficiently computable function *half* can be used as the oracle:

BREAKING RSA USING AN ORACLE

Algorithm:

for $i = 0$ to $\lceil \lg n \rceil$ **do**
  $c_i \leftarrow half(c); c \leftarrow (c \times e_k(2)) \bmod n$
$l \leftarrow 0; u \leftarrow n$
for $i = 0$ to $\lceil \lg n \rceil$ **do**
  $m \leftarrow (i + u)/2;$
  if $c_i = 1$ **then** $i \leftarrow m$ **else** $u \leftarrow m$;
output $\leftarrow [u]$

Indeed, in the first cycle

$$c_i = half(c \times (e_k(2))^i) = half(e_k(2^i w)),$$

is computed for $0 \leq i \leq \lg n$.

BREAKING RSA USING AN ORACLE

Algorithm:

for $i = 0$ to $\lceil \lg n \rceil$ **do**
  $c_i \leftarrow half(c); c \leftarrow (c \times e_k(2)) \bmod n$
$l \leftarrow 0; u \leftarrow n$
for $i = 0$ to $\lceil \lg n \rceil$ **do**
  $m \leftarrow (i + u)/2;$
  if $c_i = 1$ **then** $i \leftarrow m$ **else** $u \leftarrow m;$
output $\leftarrow [u]$

Indeed, in the first cycle

$$c_i = half(c \times (e_k(2))^i) = half(e_k(2^i w)),$$

is computed for $0 \leq i \leq \lg n$.

In the second part of the algorithm binary search is used to determine interval in which w lies. For example, we have that

$$half(e_k(w)) = 0 \equiv w \in [0, \frac{n}{2})$$

$$half(e_k(2w)) = 0 \equiv w \in [0, \frac{n}{4}) \cup [\frac{n}{2}, \frac{3n}{4})$$

$$half(e_k(4w)) = 0 \equiv w \in$$

SECURITY of RSA in PRACTICE II

There are many results for RSA showing that certain parts are as hard as whole. For example any feasible algorithm to determine the last bit of the plaintext can be converted into a feasible algorithm to determine the whole plaintext.

Example Assume that we have an algorithm H to determine whether a plaintext x designed in RSA with public key e, n is smaller than $\frac{n}{2}$ if the cryptotext y is given.

We construct an algorithm A to determine in which of the intervals $(\frac{jn}{8}, \frac{(j+1)n}{8}), 0 \leq j \leq 7$ the plaintext lies.

Basic idea H can be used to decide whether the plaintexts for cryptotexts $x^e \bmod n, 2^e x^e \bmod n, 4^e x^e \bmod n$ are smaller than $\frac{n}{2}$.

Answers

yes, yes, yes	$0 < x < \frac{n}{8}$	no, yes, yes	$\frac{n}{2} < x < \frac{5n}{8}$
yes, yes, no	$\frac{n}{8} < x < \frac{n}{4}$	no, yes, no	$\frac{5n}{8} < x < \frac{3n}{4}$
yes, no, yes	$\frac{n}{4} < x < \frac{3n}{8}$	no, no, yes	$\frac{3n}{4} < x < \frac{7n}{8}$
yes, no, no	$\frac{3n}{8} < x < \frac{n}{2}$	no, no, no	$\frac{7n}{8} < x < n$

TWO USERS SHOULD not USE THE SAME MODULUS

Otherwise, users, say A and B, would be able to decrypt messages of each other using the following method.

Decryption: B computes

$$f = \gcd(e_B d_B - 1, e_A), m = \frac{e_B d_B - 1}{f}$$

$$e_B d_B - 1 = k\phi(n) \text{ for some } k$$

It holds:

$$\gcd(e_A, \phi(n)) = 1 \Rightarrow \gcd(f, \phi(n)) = 1$$

and therefore

$$m \text{ is a multiple of } \phi(n).$$

m and e_A have no common divisor and therefore there exist integers u, v such that

$$um + ve_A = 1$$

Since m is a multiple of $\phi(n)$, we have

$$ve_A = 1 - um \equiv 1 \bmod \phi(n)$$

and since $e_A d_A \equiv 1 \bmod \phi(n)$, we have

$$(v - d_A)e_A \equiv 0 \bmod \phi(n)$$

and therefore

$$v \equiv d_A \bmod \phi(n)$$

is a decryption exponent of A. Indeed, for a cryptotext c:

$$c^v \equiv w^{e_A v} \equiv w^{e_A d_A + c\phi(n)} \equiv w \bmod (n)$$

Let a message w be encoded with a modulus n and two encryption exponents e_1 and e_2 such that gcd$(e_1, e_2) = 1$. Therefore

$$c_1 = w^{e_1} \bmod n, \qquad c_2 = w^{e_2} \bmod n;$$

Then

$$w = c_1^a c_2^b,$$

where, a, b are such that

$$a \cdot e_1 + b \cdot e_2 = 1$$

- The prime advantage of public-key cryptography is increased security – the private keys do not ever need to be transmitted or revealed to anyone.

PRIVATE-KEY versus PUBLIC-KEY CRYPTOGRAPHY

- The prime advantage of public-key cryptography is increased security – the private keys do not ever need to be transmitted or revealed to anyone.
- Public key cryptography is not meant to replace secret-key cryptography, but rather to supplement it, to make it more secure.

PRIVATE-KEY versus PUBLIC-KEY CRYPTOGRAPHY

- The prime advantage of public-key cryptography is increased security – the private keys do not ever need to be transmitted or revealed to anyone.
- Public key cryptography is not meant to replace secret-key cryptography, but rather to supplement it, to make it more secure.
- Example RSA and DES (AES) are usually combined as follows
  1. The message is encrypted with a random DES key
  2. DES-key is encrypted with RSA
  3. DES-encrypted message and RSA-encrypted DES-key are sent.

  This protocol is called RSA digital envelope.

PRIVATE-KEY versus PUBLIC-KEY CRYPTOGRAPHY

- The prime advantage of public-key cryptography is increased security – the private keys do not ever need to be transmitted or revealed to anyone.
- Public key cryptography is not meant to replace secret-key cryptography, but rather to supplement it, to make it more secure.
- Example RSA and DES (AES) are usually combined as follows
  1. The message is encrypted with a random DES key
  2. DES-key is encrypted with RSA
  3. DES-encrypted message and RSA-encrypted DES-key are sent.

  This protocol is called RSA digital envelope.
- In software (hardware) DES is generally about 100 (1000) times faster than RSA.

  If n users communicate with secrete-key cryptography, they need n (n - 1) / 2 keys.

  If n users communicate with public-key cryptography 2n keys are sufficient.

  Public-key cryptography allows spontaneous communication.

KERBEROS

We describe a very popular key distribution protocol with trusted authority TA with which each user A shares a secret key K_A.

- To communicate with user B the user A asks TA for a session key (K)
- TA chooses a random session key K, a time-stamp T, and a lifetime limit L.
- TA computes

$$m_1 = e_{K_A}(K, ID(B), T, L); \quad m_2 = e_{K_B}(K, ID(B), T, L);$$

  and sends m_1, m_2 to A.
- A decrypts m_1, recovers $K, T, L, ID(B)$, computes $m_3 = e_K(ID(B), T)$ and sends m_2 and m_3 to B.
- B decrypts m_2 and m_3, checks whether two values of T and of $ID(B)$ are the same. If so, B computes $m_4 = e_K(T + 1)$ and sends it to A.
- A decrypts m_4 and verifies that she got $T + 1$.

Part VI

Public-key cryptosystems, II. Other cryptosystems, security, PRG, hash functions

CHAPTER 6: OTHER CRYPTOSYSTEMS, PSEUDO-RANDOM NUMBER GENERATORS and HASH FUNCTIONS

A large number of interesting and important cryptosystems have already been designed. In this chapter we present several other of them in order to illustrate principles and techniques that can be used to design cryptosystems.

At first, we present several cryptosystems security of which is based on the fact that computation of square roots and discrete logarithms is in genral infeasible in some groups.

CHAPTER 6: OTHER CRYPTOSYSTEMS, PSEUDO-RANDOM NUMBER GENERATORS and HASH FUNCTIONS

A large number of interesting and important cryptosystems have already been designed. In this chapter we present several other of them in order to illustrate principles and techniques that can be used to design cryptosystems.

At first, we present several cryptosystems security of which is based on the fact that computation of square roots and discrete logarithms is in genral infeasible in some groups. Secondly, we discuss pseudo-random number generators and hash functions – other very important concepts of modern cryptography

CHAPTER 6: OTHER CRYPTOSYSTEMS, PSEUDO-RANDOM NUMBER GENERATORS and HASH FUNCTIONS

A large number of interesting and important cryptosystems have already been designed. In this chapter we present several other of them in order to illustrate principles and techniques that can be used to design cryptosystems.

At first, we present several cryptosystems security of which is based on the fact that computation of square roots and discrete logarithms is in genral infeasible in some groups. Secondly, we discuss pseudo-random number generators and hash functions – other very important concepts of modern cryptography

Finally, we discuss one of the fundamental questions of modern cryptography: when can a cryptosystem be considered as (computationally) perfectly secure?

In order to do that we will:

- discuss the role randomness play in the cryptography;
- introduce the very fundamental definitions of perfect security of cryptosystem
- present some examples of perfectly secure cryptosystems.

RABIN CRYPTOSYSTEM

Primes p, q of the form $4k+3$, so called Blum primes, are kept secret, $n = pq$ is the public key.
Encryption: of a plaintext $w < n$

$$c = w^2 \bmod n$$

RABIN CRYPTOSYSTEM

Primes p, q of the form $4k+3$, so called Blum primes, are kept secret, $n = pq$ is the public key.
Encryption: of a plaintext $w < n$

$$c = w^2 \bmod n$$

Decryption: It is easy to verify, using Euler's criterion which says that if c is a quadratic residue modulo p, then $c^{(p-1)/2} \equiv 1 \pmod p$, that

$$\pm c^{(p+1)/4} \bmod p \quad \text{and} \quad \pm c^{(q+1)/4} \bmod q$$

are two square roots of c modulo p and q. One can now obtain four square roots of c modulo n using the method shown in Appendix.

RABIN CRYPTOSYSTEM

Primes p, q of the form $4k + 3$, so called Blum primes, are kept secret, $n = pq$ is the public key.
Encryption: of a plaintext $w < n$

$$c = w^2 \bmod n$$

Decryption: It is easy to verify, using Euler's criterion which says that if c is a quadratic residue modulo p, then $c^{(p-1)/2} \equiv 1 \pmod p$, that

$$\pm c^{(p+1)/4} \bmod p \quad \text{and} \quad \pm c^{(q+1)/4} \bmod q$$

are two square roots of c modulo p and q. One can now obtain four square roots of c modulo n using the method shown in Appendix.
In case the plaintext w is a meaningful English text, it should be easy to determine w from w_1, w_2, w_3, w_4.

However, if w is a random string (say, for a key exchange) it is impossible to determine w from w_1, w_2, w_3, w_4.

RABIN CRYPTOSYSTEM

Primes p, q of the form $4k+3$, so called Blum primes, are kept secret, $n = pq$ is the public key.
Encryption: of a plaintext $w < n$

$$c = w^2 \bmod n$$

Decryption: It is easy to verify, using Euler's criterion which says that if c is a quadratic residue modulo p, then $c^{(p-1)/2} \equiv 1 \pmod{p}$, that

$$\pm c^{(p+1)/4} \bmod p \quad \text{and} \quad \pm c^{(q+1)/4} \bmod q$$

are two square roots of c modulo p and q. One can now obtain four square roots of c modulo n using the method shown in Appendix.
In case the plaintext w is a meaningful English text, it should be easy to determine w from w_1, w_2, w_3, w_4.

However, if w is a random string (say, for a key exchange) it is impossible to determine w from w_1, w_2, w_3, w_4.
Rabin did not propose this system as a practical cryptosystem.

GENERALIZED RABIN CRYPTOSYSTEM

Public key: n, B $(0 \leq B \leq n-1)$

Trapdoor: Blum primes p, q $(n = pq)$

Encryption: $e(x) = x(x+B) \bmod n$

Decryption: $d(y) = \left(\sqrt{\frac{B^2}{4} + y} - \frac{B}{2}\right) \bmod n$

GENERALIZED RABIN CRYPTOSYSTEM

Public key: n, B $(0 \leq B \leq n-1)$

Trapdoor: Blum primes p, q $(n = pq)$

Encryption: $e(x) = x(x+B) \bmod n$

Decryption: $d(y) = \left(\sqrt{\frac{B^2}{4} + y} - \frac{B}{2}\right) \bmod n$

It is easy to verify that if ω is a nontrivial square root of 1 modulo n, then there are four decryptions of $e(x)$:

$$x, \quad -x, \quad \omega\left(x + \frac{B}{2}\right) - \frac{B}{2}, \quad -\omega\left(x + \frac{B}{2}\right) - \frac{B}{2}$$

GENERALIZED RABIN CRYPTOSYSTEM

Public key: n, B $(0 \leq B \leq n-1)$

Trapdoor: Blum primes p, q $(n = pq)$

Encryption: $e(x) = x(x+B) \bmod n$

Decryption: $d(y) = \left(\sqrt{\frac{B^2}{4} + y} - \frac{B}{2}\right) \bmod n$

It is easy to verify that if ω is a nontrivial square root of 1 modulo n, then there are four decryptions of $e(x)$:

$$x, \quad -x, \quad \omega\left(x + \frac{B}{2}\right) - \frac{B}{2}, \quad -\omega\left(x + \frac{B}{2}\right) - \frac{B}{2}$$

Example

$e\left(\omega\left(x+\frac{B}{2}\right)-\frac{B}{2}\right) = \left(\omega\left(x+\frac{B}{2}\right)-\frac{B}{2}\right)\left(\omega\left(x+\frac{B}{2}\right)+\frac{B}{2}\right) = \omega^2\left(x+\frac{B}{2}\right)^2 - \left(\frac{B}{2}\right)^2 = x^2 + Bx = e(x)$

GENERALIZED RABIN CRYPTOSYSTEM

Public key: n, B $(0 \leq B \leq n-1)$

Trapdoor: Blum primes p, q $(n = pq)$

Encryption: $e(x) = x(x+B) \bmod n$

Decryption: $d(y) = \left(\sqrt{\frac{B^2}{4} + y} - \frac{B}{2}\right) \bmod n$

It is easy to verify that if ω is a nontrivial square root of 1 modulo n, then there are four decryptions of $e(x)$:

$$x, \quad -x, \quad \omega\left(x + \tfrac{B}{2}\right) - \tfrac{B}{2}, \quad -\omega\left(x + \tfrac{B}{2}\right) - \tfrac{B}{2}$$

Example

$e\left(\omega\left(x+\frac{B}{2}\right)-\frac{B}{2}\right) = \left(\omega\left(x+\frac{B}{2}\right)-\frac{B}{2}\right)\left(\omega\left(x+\frac{B}{2}\right)+\frac{B}{2}\right) = \omega^2\left(x+\frac{B}{2}\right)^2 - \left(\frac{B}{2}\right)^2 =$
$x^2 + Bx = e(x)$

Decryption of the generalized Rabin cryptosystem can be reduced to the decryption of the original Rabin cryptosystem.

Indeed, the equation $\quad x^2 + Bx \equiv y \pmod n$
can be transformed, by the substitution $x = x_1 - B/2$, *into*
$x_1^2 \equiv B^2/4 + y \pmod n$ and, by defining $c = B^2/4 + y$, *into* ${x_1}^2 \equiv c \pmod n$
Decryption can be done by factoring n and solving congruences

$${x_1}^2 \equiv c \pmod p \qquad {x_1}^2 \equiv c \pmod q$$

SECURITY of RABIN CRYPTOSYSTEM

We show that any hypothetical decryption algorithm **A** for Rabin cryptosystem, can be used, as an oracle, in the following Las Vegas algorithm, to factor an integer n.

Algorithm:

1. Choose a random $r, 1 \leq r \leq n-1$;
2. Compute $y = (r^2 - B^2/4) \bmod n$; $\{y = e_k(r - B/2)\}$.
3. Call $A(y)$, to obtain a decryption $x = \left(\sqrt{\frac{B^2}{4} + y} - \frac{B}{2}\right) \bmod n$;
4. Compute $x_1 = x + B/2$; $\{x_1^2 \equiv r^2 \bmod n\}$
5. **if** $x_1 = \pm r$ **then quit** (failure)
   else $\gcd(x_1 + r, n) = p$ or q

SECURITY of RABIN CRYPTOSYSTEM

We show that any hypothetical decryption algorithm **A** for Rabin cryptosystem, can be used, as an oracle, in the following Las Vegas algorithm, to factor an integer n.

Algorithm:

1. Choose a random $r, 1 \leq r \leq n-1$;
2. Compute $y = (r^2 - B^2/4) \bmod n$; $\quad \{y = e_k(r - B/2)\}$.
3. Call $A(y)$, to obtain a decryption $x = \left(\sqrt{\frac{B^2}{4} + y} - \frac{B}{2}\right) \bmod n$;
4. Compute $x_1 = x + B/2$; $\quad \{x_1^2 \equiv r^2 \bmod n\}$
5. **if** $x_1 = \pm r$ **then quit** (failure)
   else $\gcd(x_1 + r, n) = p$ or q

Indeed, after Step 4, either $x_1 = \pm r \bmod n$ or $x_1 = \pm \omega r \bmod n$.
In the second case we have

$$n \,|\, (x_1 - r)(x_1 + r),$$

but n does not divide either factor $x_1 - r$ or $x_1 + r$.
Therefore computation of $\gcd(x_1 + r, n)$ or $\gcd(x_1 - r, n)$ must yield factors of n.

ElGamal CRYPTOSYSTEM

Design: choose a large prime p – (with at least 150 digits).
choose two random integers $1 \leq q, x < p$ – where q is a primitive element of $Z^*{}_p$
calculate $y = q^x \bmod p$.

ElGamal CRYPTOSYSTEM

Design: choose a large prime p – (with at least 150 digits).
choose two random integers $1 \leq q, x < p$ – where q is a primitive element of $Z^*{}_p$
calculate $y = q^x \bmod p$.

Public key: p, q, y;   trapdoor: x

ElGamal CRYPTOSYSTEM

Design: choose a large prime p – (with at least 150 digits).
choose two random integers $1 \leq q, x < p$ – where q is a primitive element of $Z^*{}_p$
calculate $y = q^x \bmod p$.

Public key: p, q, y; trapdoor: x

Encryption of a plaintext w: choose a random r and compute

$$a = q^r \bmod p, \qquad b = y^r w \bmod p$$

Cryptotext: $c = (a, b)$

(Cryptotext contains indirectly r and the plaintext is "masked" by multiplying with y^r (and taking modulo p))

ElGamal CRYPTOSYSTEM

Design: choose a large prime p – (with at least 150 digits).
choose two random integers $1 \leq q, x < p$ – where q is a primitive element of Z^*_p
calculate $y = q^x \bmod p$.

Public key: p, q, y; trapdoor: x

Encryption of a plaintext w: choose a random r and compute

$$a = q^r \bmod p, \qquad b = y^r w \bmod p$$

Cryptotext: $c = (a, b)$
(Cryptotext contains indirectly r and the plaintext is "masked" by multiplying with y^r (and taking modulo p))

Decryption: $w = \frac{b}{a^x} \bmod p = ba^{-x} \bmod p$.

Proof of correctness: $a^x \equiv q^{rx} \bmod p$

$$\frac{b}{a^x} \equiv \frac{y^r w}{a^x} \equiv \frac{q^{rx} w}{q^{rx}} \equiv w (\bmod p)$$

ElGamal CRYPTOSYSTEM

Design: choose a large prime p – (with at least 150 digits).
choose two random integers $1 \leq q, x < p$ – where q is a primitive element of Z^*_p
calculate $y = q^x \bmod p$.

Public key: p, q, y; trapdoor: x

Encryption of a plaintext w: choose a random r and compute

$$a = q^r \bmod p, \qquad b = y^r w \bmod p$$

Cryptotext: $c = (a, b)$
(Cryptotext contains indirectly r and the plaintext is "masked" by multiplying with y^r (and taking modulo p))

Decryption: $w = \frac{b}{a^x} \bmod p = ba^{-x} \bmod p$.

Proof of correctness: $a^x \equiv q^{rx} \bmod p$

$$\frac{b}{a^x} \equiv \frac{y^r w}{a^x} \equiv \frac{q^{rx} w}{q^{rx}} \equiv w (\bmod p)$$

Note: Security of the ElGamal cryptosystem is based on infeasibility of the discrete logarithm computation.

SHANKS' ALGORITHM for DISCRETE ALGORITHM

Let $m = \lceil \sqrt{(p-1)} \rceil$. The following algorithm computes $\lg_q y$ in Z^*_p.

1. Compute $q^{mj} \bmod p$, $0 \leq j \leq m-1$.
2. Create list L_1 of m pairs $(j, q^{mj} \bmod p)$, sorted by the second item.
3. Compute $yq^{-i} \bmod p$, $0 \leq i \leq m-1$.
4. Create list L_2 of pairs $(i, yq^{-i} \bmod p)$ sorted by the second item.
5. Find two pairs, one $(j, z) \in L_1$ and second $(i, z) \in L_2$

SHANKS' ALGORITHM for DISCRETE ALGORITHM

Let $m = \lceil \sqrt{(p-1)} \rceil$. The following algorithm computes $\lg_q y$ in $Z^*{}_p$.

1. Compute $q^{mj} \bmod p$, $\quad 0 \leq j \leq m-1$.
2. Create list L_1 of m pairs $(j, q^{mj} \bmod p)$, sorted by the second item.
3. Compute $yq^{-i} \bmod p$, $\quad 0 \leq i \leq m-1$.
4. Create list L_2 of pairs $(i, yq^{-i} \bmod p)$ sorted by the second item.
5. Find two pairs, one $(j, z) \in L_1$ and second $(i, z) \in L_2$

If such a search is successful, then

$$q^{mj} \bmod p = z = yq^{-i} \bmod p$$

and as the result

$$\lg_q y \equiv (mj + i) \bmod (p-1).$$

Therefore

$$q^{mj+i} \equiv y \pmod p$$

On the other hand, for any y we can write

$$\lg_q y = mj + i,$$

For some $0 \leq i, j \leq m-1$. Hence the search in the Step 5 of the algorithm has to be successful.

BIT SECURITY of DISCRETE LOGARITHM

Let us consider problem to compute $L_i(y) = i$-th least significant bit of $\lg_q y$ in Z^*_p.

Result 1 $L_1(y)$ can be computed efficiently.
To show that we use the fact that the set $QR(p)$ has $(p-1)/2$ elements.
Let q be a primitive element of Z^*_p. Clearly, $q^a \in QR(p)$ if a is even. Since the elements

$$q^0 \bmod p, q^2 \bmod p, \ldots, q^{p-3} \bmod p$$

are all distinct, we have that

$$QR(p) = \{q^{2i} \bmod p \mid 0 \leq i \leq (p-3)/2\}$$

Consequence: y is a quadratic residue iff $\lg_q y$ is even, that is iff $L_1(y) = 0$.

By Euler's criterion y is a quadratic residue if $y^{(p-1)/2} \equiv 1 \bmod p$
$L_1(y)$ can therefore be computed as follows:

$$\begin{array}{ll} L_1(y) = 0 & \text{if } y^{(p-1)/2} \equiv 1 \bmod p; \\ L_1(y) = 1 & \text{otherwise} \end{array}$$

BIT SECURITY of DISCRETE LOGARITHM

Let us consider problem to compute $L_i(y) = i$-th least significant bit of $\lg_q y$ in $Z^*{}_p$.

Result 1 $L_1(y)$ can be computed efficiently.
To show that we use the fact that the set $QR(p)$ has $(p-1)/2$ elements.
Let q be a primitive element of $Z^*{}_p$. Clearly, $q^a \in QR(p)$ if a is even. Since the elements

$$q^0 \bmod p, q^2 \bmod p, \ldots, q^{p-3} \bmod p$$

are all distinct, we have that

$$QR(p) = \{q^{2i} \bmod p \,|\, 0 \leq i \leq (p-3)/2\}$$

Consequence: y is a quadratic residue iff $\lg_q y$ is even, that is iff $L_1(y) = 0$.

By Euler's criterion y is a quadratic residue if $y^{(p-1)/2} \equiv 1 \bmod p$
$L_1(y)$ can therefore be computed as follows:

$$\begin{array}{ll} L_1(y) = 0 & \text{if } y^{(p-1)/2} \equiv 1 \bmod p; \\ L_1(y) = 1 & \text{otherwise} \end{array}$$

Result 2 Efficient computability of $L_i(y), i > 1$ in $Z^*{}_p$ would imply efficient computability of the discrete logarithm in $Z^*{}_p$.

GROUP VERSION of ElGamal CRYPTOSYSTEM

A group version of discrete logarithm probem

Given a group $(G, \circ)$, $\alpha \in G$, $\beta \in \{\alpha^i \mid i \geq 0\}$. Find

$$\log_\alpha \beta = k \text{ such that } \alpha^k = \beta$$

GROUP VERSION of ElGamal CRYPTOSYSTEM

ElGamal cryptosystem can be implemented in any group in which discrete logarithm problem is infeasible.

Cryptosystem for $(G, \circ)$
Public key: α, β
Trapdoor: k such that $\alpha^k = \beta$

Encryption: of a plaintext w and a random integer k

$$e(w, k) = (y_1, y_2) \text{ where } y_1 = \alpha^k, y_2 = w \circ \beta^k$$

Decryption: of cryptotext (y_1, y_2):

$$d(y_1, y_2) = y_2 \circ y_1^{-k}$$

An important special case is that of computation of discrete logarithm in a group of points of an eliptic curve defined over a finite field.

WILLIAMS CRYPTOSYSTEM – BASICS

This cryptosystem is similar to RSA, but with number operations performed in a quadratic field. Complexity of the cryptanalysis of the Williams cryptosystem is equivalent to factoring.

Consider numbers of the form

$$\alpha = a + b\sqrt{c}$$

where a, b, c are integers.
If c is fixed, α can be viewed as a pair (a, b).

$$\alpha_1 + \alpha_2 = (a_1, b_1) + (a_2, b_2) = (a_1 + a_2, b_1 + b_2)$$

$$\alpha_1\alpha_2 = (a_1, b_1) \cdot (a_2, b_2) = (a_1 a_2 + \mathrm{c}\, b_1 b_2, a_1 b_2 + b_1 a_2)$$

The conjugate $\overline{\alpha}$ of α of a is defined by

$$\overline{\alpha} = a - b\sqrt{c}$$

WILLIAMS CRYPTOSYSTEM – BASICS

This cryptosystem is similar to RSA, but with number operations performed in a quadratic field. Complexity of the cryptanalysis of the Williams cryptosystem is equivalent to factoring.

Consider numbers of the form

$$\alpha = a + b\sqrt{c}$$

where a, b, c are integers.
If c is fixed, α can be viewed as a pair (a, b).

$$\alpha_1 + \alpha_2 = (a_1, b_1) + (a_2, b_2) = (a_1 + a_2, b_1 + b_2)$$

$$\alpha_1\alpha_2 = (a_1, b_1) \cdot (a_2, b_2) = (a_1 a_2 + \mathrm{c}\, b_1 b_2, a_1 b_2 + b_1 a_2)$$

The conjugate $\overline{\alpha}$ of α of a is defined by

$$\overline{\alpha} = a - b\sqrt{c}$$

Auxiliary functions:

$$X_i(\alpha) = \frac{\alpha^i + \alpha^{-i}}{2}$$

$$Y_i(\alpha) = \frac{b(\alpha^i - \alpha^{-i})}{(\alpha - \overline{\alpha})} \left(= \frac{\alpha - \overline{\alpha}^i}{2\sqrt{c}} \right)$$

Hence

$$\alpha^i = X_i(\alpha) + Y_i(\alpha)\sqrt{c}$$
$$\overline{\alpha}^i = X_i(\alpha) - Y_i(\alpha)\sqrt{c}$$

Assume now

$$a^2 - cb^2 = 1$$

Then $\alpha\overline{\alpha} = 1$ and consequently

$$X_I{}^2 - cY_I{}^2 = 1$$

Moreover, for $j \geq i$

$$X_{I+J} = 2X_I X_J + X_{J-1}$$
$$Y_{I+J} = 2Y_I X_J + Y_{J-1}$$

From these and following equations:

$$X_{I+J} = 2X_I X_J + cY_I Y_J$$
$$Y_{I+J} = 2Y_I X_J + X_I Y_J$$

we get the recursive formulas:

$$X_{2i} = X_i{}^2 + cY_i{}^2 = 2X_i{}^2 - 1$$
$$Y_{2i} = 2X_i Y_i$$
$$X_{2i+1} = 2X_i Y_{i+1} - X_1$$
$$Y_{2i+1} = 2X_i Y_{i+1} - Y_1$$

Consequences: 1. X_i and Y_i can be, given i, computed fast.
Remark Since $X_0 = 1, X_1 = a, X_i$ does not depend on b.

WHEN is a CRYPTOSYSTEM (perfectly) SECURE?

First question: Is it enough for perfect security of a cryptosystem that one cannot get a plaintext from a cryptotext?

WHEN is a CRYPTOSYSTEM (perfectly) SECURE?

First question: Is it enough for perfect security of a cryptosystem that one cannot get a plaintext from a cryptotext?

NO, NO, NO
WHY

WHEN is a CRYPTOSYSTEM (perfectly) SECURE?

First question: Is it enough for perfect security of a cryptosystem that one cannot get a plaintext from a cryptotext?

NO, NO, NO
WHY

For many applications it is crucial that no information about the plaintext could be obtained.

- Intuitively, a cryptosystem is (perfectly) secure if one cannot get any (new) information about the corresponding plaintext from any cryptotext.
- It is very nontrivial to define fully precisely when a cryptosystem is (computationally) perfectly secure.
- It has been shown that perfectly secure cryptosystems have to use randomized encryptions.

Randomness and cryptography are deeply related.

CRYPTOGRAPHY and RANDOMNESS

Randomness and cryptography are deeply related.

1. Prime goal of any good encryption method is to transform even a highly nonrandom plaintext into a highly random cryptotext. (Avalanche effect.)

CRYPTOGRAPHY and RANDOMNESS

Randomness and cryptography are deeply related.

1. Prime goal of any good encryption method is to transform even a highly nonrandom plaintext into a highly random cryptotext. (Avalanche effect.)

   Example Let e_k be an encryption algorithm, x_0 be a plaintext. And

   $$x_i = e_k(x_{i-1}), i \geq 1.$$

   It is intuitively clear that if encryption e_k is "cryptographically secure", then it is very, very likely that the sequence $x_0\, x_1\, x_2\, x_3$ is (quite) random.

   Perfect encryption should therefore produce (quite) perfect (pseudo)randomness.

CRYPTOGRAPHY and RANDOMNESS

Randomness and cryptography are deeply related.

1. Prime goal of any good encryption method is to transform even a highly nonrandom plaintext into a highly random cryptotext. (Avalanche effect.)

   Example Let e_k be an encryption algorithm, x_0 be a plaintext. And
   $$x_i = e_k(x_{i-1}), i \geq 1.$$
   It is intuitively clear that if encryption e_k is "cryptographically secure", then it is very, very likely that the sequence $x_0\ x_1\ x_2\ x_3$ is (quite) random.

   Perfect encryption should therefore produce (quite) perfect (pseudo)randomness.

2. The other side of the relation is more complex. It is clear that perfect randomness together with ONE-TIME PAD cryptosystem produces perfect secrecy. The price to pay: a key as long as plaintext is needed.

   The way out seems to be to use an encryption algorithm with a pseudo-random generator to generate a long pseudo-random sequence from a short seed and to use the resulting sequence with ONE-TIME PAD.

CRYPTOGRAPHY and RANDOMNESS

Randomness and cryptography are deeply related.

1. Prime goal of any good encryption method is to transform even a highly nonrandom plaintext into a highly random cryptotext. (Avalanche effect.)

   Example Let e_k be an encryption algorithm, x_0 be a plaintext. And

   $$x_i = e_k(x_{i-1}), i \geq 1.$$

   It is intuitively clear that if encryption e_k is "cryptographically secure", then it is very, very likely that the sequence $x_0\ x_1\ x_2\ x_3$ is (quite) random.

   Perfect encryption should therefore produce (quite) perfect (pseudo)randomness.

2. The other side of the relation is more complex. It is clear that perfect randomness together with ONE-TIME PAD cryptosystem produces perfect secrecy. The price to pay: a key as long as plaintext is needed.

   The way out seems to be to use an encryption algorithm with a pseudo-random generator to generate a long pseudo-random sequence from a short seed and to use the resulting sequence with ONE-TIME PAD.

   Basic question: When is a pseudo-random generator good enough for cryptographical purposes?

SECURE ENCRYPTIONS – BASIC CONCEPTS I

We now start to discuss a very nontrivial question: when is an encryption scheme computationally perfectly SECURE?

SECURE ENCRYPTIONS – BASIC CONCEPTS I

We now start to discuss a very nontrivial question: when is an encryption scheme computationally perfectly SECURE?

At first, we introduce two very basic technical concepts:

Definition A function f:$N \rightarrow R$ is a negligible function if for any polynomial $p(n)$ and for almost all n:

$$f(n) \leq \frac{1}{p(n)}$$

SECURE ENCRYPTIONS – BASIC CONCEPTS I

We now start to discuss a very nontrivial question: when is an encryption scheme computationally perfectly SECURE?

At first, we introduce two very basic technical concepts:

Definition A function f:$N \to R$ is a negligible function if for any polynomial $p(n)$ and for almost all n:

$$f(n) \leq \frac{1}{p(n)}$$

Definition – computational distinguishability Let $X = \{X_n\}_{n \in N}$ and $Y = \{Y_n\}_{n \in N}$ be probability ensembles such that each X_n and Y_n ranges over strings of length n. We say that X and Y are computationally indistinguishable if for every feasible algorithm A the difference

$$d_A(n) = | Pr[A(X_n) = 1] - Pr[A(Y_n) = 1] |$$

is a negligible function in n.

SECURE ENCRYPTIONS – PSEUDORANDOM GENERATORS

In cryptography random sequences can be usually be well enough replaced by pseudorandom sequences generated by (cryptographically perfect) pseudorandom generators.

In cryptography random sequences can be usually be well enough replaced by pseudorandom sequences generated by (cryptographically perfect) pseudorandom generators.

Definition - pseudorandom generator. Let $l(n) : N \rightarrow N$ be such that $l(n) > n$ for all n. A (computationally indistinguishable) pseudorandom generator with a stretch function l, is an efficient deterministic algorithm which on the input of a random n-bit seed outputs a $l(n)$-bit sequence which is computationally indistinguishable from any random $l(n)$-bit sequence.

SECURE ENCRYPTIONS – PSEUDORANDOM GENERATORS

In cryptography random sequences can be usually be well enough replaced by pseudorandom sequences generated by (cryptographically perfect) pseudorandom generators.

Definition - pseudorandom generator. Let $l(n) : N \rightarrow N$ be such that $l(n) > n$ for all n. A (computationally indistinguishable) pseudorandom generator with a stretch function l, is an efficient deterministic algorithm which on the input of a random n-bit seed outputs a $l(n)$-bit sequence which is computationally indistinguishable from any random $l(n)$-bit sequence.

Theorem Let f be a one-way function which is length preserving and efficiently computable, and b be a hard core predicate of f, then

$$G(s) = b(s) \cdot b(f(s)) \cdots b\left(f^{l(|s|)-1}(s)\right)$$

is a (computationally indistinguishable) pseudorandom generator with stretch function $l(n)$.

SECURE ENCRYPTIONS – PSEUDORANDOM GENERATORS

In cryptography random sequences can be usually be well enough replaced by pseudorandom sequences generated by (cryptographically perfect) pseudorandom generators.

Definition - pseudorandom generator. Let $l(n) : N \to N$ be such that $l(n) > n$ for all n. A (computationally indistinguishable) pseudorandom generator with a stretch function l, is an efficient deterministic algorithm which on the input of a random n-bit seed outputs a $l(n)$-bit sequence which is computationally indistinguishable from any random $l(n)$-bit sequence.

Theorem Let f be a one-way function which is length preserving and efficiently computable, and b be a hard core predicate of f, then

$$G(s) = b(s) \cdot b(f(s)) \cdots b\left(f^{l(|s|)-1}(s)\right)$$

is a (computationally indistinguishable) pseudorandom generator with stretch function $l(n)$.

Definition A predicate b is a hard core predicate of the function f if b is easy to evaluate, but $b(x)$ is hard to predict from $f(x)$. (That is, it is unfeasible, given $f(x)$ where x is uniformly chosen, to predict $b(x)$ substantially better than with the probability 1/2.)

It is conjectured that the least significant bit of the modular squaring function $x^2 \bmod n$ is a hard-core predicate.

SECURE ENCRYPTIONS – PSEUDORANDOM GENERATORS

In cryptography random sequences can be usually be well enough replaced by pseudorandom sequences generated by (cryptographically perfect) pseudorandom generators.

Definition - pseudorandom generator. Let $l(n) : N \to N$ be such that $l(n) > n$ for all n. A (computationally indistinguishable) pseudorandom generator with a stretch function l, is an efficient deterministic algorithm which on the input of a random n-bit seed outputs a $l(n)$-bit sequence which is computationally indistinguishable from any random $l(n)$-bit sequence.

Theorem Let f be a one-way function which is length preserving and efficiently computable, and b be a hard core predicate of f, then

$$G(s) = b(s) \cdot b(f(s)) \cdots b\left(f^{l(|s|)-1}(s)\right)$$

is a (computationally indistinguishable) pseudorandom generator with stretch function $l(n)$.

Definition A predicate b is a hard core predicate of the function f if b is easy to evaluate, but $b(x)$ is hard to predict from f(x). (That is, it is unfeasible, given f(x) where x is uniformly chosen, to predict $b(x)$ substantially better than with the probability 1/2.)

It is conjectured that the least significant bit of the modular squaring function $x^2 \bmod n$ is a hard-core predicate.

CRYPTOGRAPHICALLY STRONG PSEUDO-RANDOM GENERATORS

Fundamental question: when is a pseudo-random generator good enough for cryptographical purposes?

Basic concept: A pseudo-random generator is called cryptographically strong if the sequence of bits it produces, from a short random seed, is so good for using with ONE-TIME PAD cryptosystem, that no polynomial time algorithm allows a cryptanalyst to learn any information about the plaintext from the cryptotext.

A cryptographically strong pseudo-random generator would therefore provide sufficient security in a secret-key cryptosystem if both parties agree on some short seed and never use it twice.

As discussed later: Cryptographically strong pseudo-random generators could provide perfect secrecy also for public-key cryptography.

CRYPTOGRAPHICALLY STRONG PSEUDO-RANDOM GENERATORS

Fundamental question: when is a pseudo-random generator good enough for cryptographical purposes?

Basic concept: A pseudo-random generator is called cryptographically strong if the sequence of bits it produces, from a short random seed, is so good for using with ONE-TIME PAD cryptosystem, that no polynomial time algorithm allows a cryptanalyst to learn any information about the plaintext from the cryptotext.

A cryptographically strong pseudo-random generator would therefore provide sufficient security in a secret-key cryptosystem if both parties agree on some short seed and never use it twice.

As discussed later: Cryptographically strong pseudo-random generators could provide perfect secrecy also for public-key cryptography.

Problem: Do cryptographically strong pseudo-random generators exist?

CRYPTOGRAPHICALLY STRONG PSEUDO-RANDOM GENERATORS

Fundamental question: when is a pseudo-random generator good enough for cryptographical purposes?

Basic concept: A pseudo-random generator is called cryptographically strong if the sequence of bits it produces, from a short random seed, is so good for using with ONE-TIME PAD cryptosystem, that no polynomial time algorithm allows a cryptanalyst to learn any information about the plaintext from the cryptotext.

A cryptographically strong pseudo-random generator would therefore provide sufficient security in a secret-key cryptosystem if both parties agree on some short seed and never use it twice.

As discussed later: Cryptographically strong pseudo-random generators could provide perfect secrecy also for public-key cryptography.

Problem: Do cryptographically strong pseudo-random generators exist?

Remark: The concept of a cryptographically strong pseudo-random generator is one of the key concepts of the foundations of computing.

Indeed, a cryptographically strong pseudo-random generator exists if and only if a one-way function exists what is equivalent with $P \neq UP$ and what implies $P \neq NP$.

CANDIDATES for CRYPTOGRAPHICALLY STRONG PSEUDO-RANDOM GENERATORS

So far there are only candidates for cryptographically strong pseudo-random generators.

CANDIDATES for CRYPTOGRAPHICALLY STRONG PSEUDO-RANDOM GENERATORS

So far there are only candidates for cryptographically strong pseudo-random generators.

For example, cryptographically strong are all pseudo-random generators that are unpredictable to the left in the sense that a cryptanalyst that knows the generator and sees the whole generated sequence except its first bit has no better way to find out this first bit than to toss the coin.

CANDIDATES for CRYPTOGRAPHICALLY STRONG PSEUDO-RANDOM GENERATORS

So far there are only candidates for cryptographically strong pseudo-random generators.

For example, cryptographically strong are all pseudo-random generators that are unpredictable to the left in the sense that a cryptanalyst that knows the generator and sees the whole generated sequence except its first bit has no better way to find out this first bit than to toss the coin.

It has been shown that if integer factoring is intractable, then the so-called *BBS* pseudo-random generator, discussed below, is unpredictable to the left.

(We make use of the fact that if factoring is unfeasible, then for almost all quadratic residues x mod n, coin-tossing is the best possible way to estimate the least significant bit of x after seeing x^2 mod n.)

CANDIDATES for CRYPTOGRAPHICALLY STRONG PSEUDO-RANDOM GENERATORS

So far there are only candidates for cryptographically strong pseudo-random generators.

For example, cryptographically strong are all pseudo-random generators that are unpredictable to the left in the sense that a cryptanalyst that knows the generator and sees the whole generated sequence except its first bit has no better way to find out this first bit than to toss the coin.

It has been shown that if integer factoring is intractable, then the so-called *BBS* pseudo-random generator, discussed below, is unpredictable to the left.

(We make use of the fact that if factoring is unfeasible, then for almost all quadratic residues x mod n, coin-tossing is the best possible way to estimate the least significant bit of x after seeing x^2 mod n.)

Let n be a Blum integer. Choose a random quadratic residue x_0 (modulo n).

For $i \geq 0$ let

$$x_{i+1} = {x_i}^2 \text{mod } n, b_i = \text{the least significant bit of } x_l$$

For each integer i, let

$$BBS_{n,i}(x_0) = b_0 \dots b_{i-1}$$

be the first i bits of the pseudo-random sequence generated from the seed x_0 by the *BBS* pseudo-random generator.

BBS PSEUDO-RANDOM GENERATOR – ANALYSIS

Choose random x, relatively prime to n, compute $x_0 = x^2 \bmod n$
Let $x_{i+1} = {x_i}^2 \bmod n$, and b_i be the least significant bit of x_i
$$BBS_{n,i}(x_0) = b_0 \dots b_{i-1}$$

BBS PSEUDO-RANDOM GENERATOR – ANALYSIS

Choose random x, relatively prime to n, compute $x_0 = x^2 \bmod n$
Let $x_{i+1} = {x_i}^2 \bmod n$, and b_i be the least significant bit of x_i
$$BBS_{n,i}(x_0) = b_0 \dots b_{i-1}$$

Assume that the pseudo-random generator BBS with a Blum integer is not unpredictable to the left.

Let y be a quadratic residue from ${Z_n}^*$.

Compute $BBS_{n,i-1}(y)$ for some $i > 1$.

Let us pretend that last $(i-1)$ bits of $BBS_{n,i}(x)$ are actually the first $(i-1)$ bits of $BBS_{n,i-1}(y)$, where x is the principal square root of y.

Hence, if the BBS pseudo-random generator is not unpredictable to the left, then there exists a better method than coin-tossing to determine the least significant bit of x, what is, as mentioned above, impossible.

RANDOMIZED ENCRYPTIONS

From security point of view, public-key cryptography with deterministic encryptions has the following serious drawback:

A cryptoanalyst who knows the public encryption function e_k and a cryptotext c can try to guess a plaintext w, compute $e_k(w)$ and compare it with c.

The purpose of randomized encryptions is to encrypt messages, using randomized algorithms, in such a way that one can prove that no feasible computation on the cryptotext can provide any information whatsoever about the corresponding plaintext (except with a negligible probability).

RANDOMIZED ENCRYPTIONS

From security point of view, public-key cryptography with deterministic encryptions has the following serious drawback:

A cryptoanalyst who knows the public encryption function e_k and a cryptotext c can try to guess a plaintext w, compute $e_k(w)$ and compare it with c.

The purpose of randomized encryptions is to encrypt messages, using randomized algorithms, in such a way that one can prove that no feasible computation on the cryptotext can provide any information whatsoever about the corresponding plaintext (except with a negligible probability).

Formal setting: Given:

plaintext-space	P
cryptotext	C
key-space	K
random-space	R

encryption: $e_k : P \times R \rightarrow C$
decryption: $d_k : C \rightarrow P$ *or* $C \rightarrow 2^P$ such that for any p, r:

$$d_k(e_k(p, r)) = p.$$

- d_k, e_k should be easy to compute.
- Given e_k, it should be unfeasible to determine d_k.

Definition – semantic security of encryption A cryptographic system is semantically secure if for every feasible algorithm A, there exists a feasible algorithm B so that for every two functions

$$f, h : \{0,1\}^* \rightarrow \{0,1\}^n$$

and all probability ensembles $\{X_n\}_{n \in N}$, where X_n ranges over $\{0,1\}^n$

$$Pr[A(E(X_n), h(X_n)) = f(X_n)] < Pr[B(h(X_n)) = f(X_n)] + \mu(n),$$

where μ is a negligible function.

Definition – semantic security of encryption A cryptographic system is semantically secure if for every feasible algorithm A, there exists a feasible algorithm B so that for every two functions

$$f, h : \{0,1\}^* \rightarrow \{0,1\}^n$$

and all probability ensembles $\{X_n\}_{n \in N}$, where X_n ranges over $\{0,1\}^n$

$$Pr[A(E(X_n), h(X_n)) = f(X_n)] < Pr[B(h(X_n)) = f(X_n)] + \mu(n),$$

where μ is a negligible function.

It can be shown that any semantically secure public-key cryptosystem must use a randomized encryption algorithm.

RSA cryptosystem is not secure in the above sense. However, randomized versions of RSA are semantically secure.

SECURE ENCRYPTIONS – SECOND DEFINITION

Definition A randomized-encryption cryptosystem is **polynomial time** secure if, for any c $\in N$ and sufficiently large s $\in N$ (security parameter), any randomized polynomial time algorithms that takes as input s (in unary) and the public key, cannot distinguish between randomized encryptions, by that key, of two given messages of length c, with the probability larger than $\frac{1}{2} + \frac{1}{s^c}$.

SECURE ENCRYPTIONS – SECOND DEFINITION

Definition A randomized-encryption cryptosystem is **polynomial time** secure if, for any c $\in N$ and sufficiently large s $\in N$ (security parameter), any randomized polynomial time algorithms that takes as input s (in unary) and the public key, cannot distinguish between randomized encryptions, by that key, of two given messages of length c, with the probability larger than $\frac{1}{2} + \frac{1}{s^c}$.

Both definitions are equivalent.

Definition A randomized-encryption cryptosystem is **polynomial time** secure if, for any $c \in N$ and sufficiently large $s \in N$ (security parameter), any randomized polynomial time algorithms that takes as input s (in unary) and the public key, cannot distinguish between randomized encryptions, by that key, of two given messages of length c, with the probability larger than $\frac{1}{2} + \frac{1}{s^c}$.

Both definitions are equivalent.

Example of a polynomial-time secure randomized (Bloom-Goldwasser) encryption:

p, q - large Blum primes $n = p \times q$ - key
Plaintext-space - all binary strings
Random-space – QR_n
Crypto-space - $QR_n \times \{0, 1\}^*$

SECURE ENCRYPTIONS – SECOND DEFINITION

Definition A randomized-encryption cryptosystem is **polynomial time** secure if, for any c $\in N$ and sufficiently large s $\in N$ (security parameter), any randomized polynomial time algorithms that takes as input s (in unary) and the public key, cannot distinguish between randomized encryptions, by that key, of two given messages of length c, with the probability larger than $\frac{1}{2} + \frac{1}{s^c}$.

Both definitions are equivalent.

Example of a polynomial-time secure randomized (Bloom-Goldwasser) encryption:

p, q - large Blum primes n = p $\times$ q - key
Plaintext-space - all binary strings
Random-space – QR_n
Crypto-space - $QR_n \times \{0, 1\}^*$

Encryption: Let w be a t-bit plaintext and x_0 a random quadratic residue modulo n. Compute x_t and $BBS_{n,t}(x_0)$ using the recurrence

$$x_{i+1} = x_i^2 \mod n$$

Cryptotext: $(x_t, w \oplus BBS_{n,t}(x_0))$

SECURE ENCRYPTIONS – SECOND DEFINITION

Definition A randomized-encryption cryptosystem is **polynomial time** secure if, for any c $\in N$ and sufficiently large s $\in N$ (security parameter), any randomized polynomial time algorithms that takes as input s (in unary) and the public key, cannot distinguish between randomized encryptions, by that key, of two given messages of length c, with the probability larger than $\frac{1}{2} + \frac{1}{s^c}$.

Both definitions are equivalent.

Example of a polynomial-time secure randomized (Bloom-Goldwasser) encryption:

p, q - large Blum primes n = p $\times$ q - key
Plaintext-space - all binary strings
Random-space – QR_n
Crypto-space - $QR_n \times \{0,1\}^*$

Encryption: Let w be a t-bit plaintext and x_0 a random quadratic residue modulo n. Compute x_t and $BBS_{n,t}(x_0)$ using the recurrence

$$x_{i+1} = x_i^2 \mod n$$

Cryptotext: $(x_t, w \oplus BBS_{n,t}(x_0))$
Decryption: Legal user, knowing p, q, can compute x_0 from x_t, then $BBS_{n,t}(x_0)$, and finally w.

Another very simple, fundamental and important cryptographic concept is that of hash functions.

Hash functions

$$h : \{0,1\}^* \rightarrow \{0,1\}^m; \qquad h : \{0,1\}^n \rightarrow \{0,1\}^m,\ n >> m$$

map (very) long messages w into short ones, called usually messages digests or hashes or fingerprints of w, in a way that has important cryptographic properties.

HASH FUNCTIONS

Another very simple, fundamental and important cryptographic concept is that of hash functions.

Hash functions

$$h : \{0,1\}^* \rightarrow \{0,1\}^m; \qquad h : \{0,1\}^n \rightarrow \{0,1\}^m,\ n >> m$$

map (very) long messages w into short ones, called usually messages digests or hashes or fingerprints of w, in a way that has important cryptographic properties.

Digital signatures are one of important applications of hash functions.

In most of the digital signature schemes, to be discussed in the next chapter, the length of a signature is at least as long as of the message being signed. This is clearly a big disadvantage.

To remedy this situation, signing procedure is applied to a hash of the message, rather than to the message itself. This is OK provided the hash function has good cryptographic properties, discussed next.

We now derive basic properties cryptographically good hash functions should have – by analysing several possible attacks on their use.

Attack 1 If Eve gets a valid signature (w,y), where $y = sig_k(h(w))$ and she would be able to find w' such that h(w')=h(w), then also (w',y), a forgery, would be a valid signature.

Cryptographically good hash function should therefore have the following weak collision-free property

Definition 1. Let w be a message. A hash function h is weakly collision-free for w, if it is computationally infeasible to find a w' such that h(w)=h(w').

PROPERTIES GOOD HASH FUNCTIONS SHOULD HAVE II.

Attack 2 If Eve finds two w and w' such that h(w')=h(w), she can ask Alice to sign h(w) to get signature s and then Eve can create a forgery (w',s).

Cryptographically good hash function should therefore have the following strong collision-free property

Definition 2. A hash function h is strongly collision-free if it is computationally infeasible to find two elements $w \neq w'$ such that h(w)=h(w').

PROPERTIES HASH FUNCTIONS SHOULD HAVE III.

Attack 3 If Eve can compute signature s of a random z, and then she can find w such that z=h(w), then Eve can create forgery (w,s).

To exclude such an attack, hash functions should have the following one-wayness property.

Definition 3. A hash function h is one-way if it is computationally infeasible to find, given z, an w such that h(w)=z.

One can show that if a hash function has strongly collision-free property, then it has one-wayness property.

An important use of hash functions is to protect integrity of data in the following way:

The problem of protecting data of arbitrary length is reduced, using hash functions, to the problem to protect integrity of the data of fixed (and small) length – of their fingerprints.

In addition, to send reliably a message w through an unreliable (and cheap) channel, one sends also its (small) hash h(w) through a very secure (and therefore expensive) channel.

The receiver, familiar also with the hash function h that is being used, can then verify the integrity of the message w' he receives by computing h(w') and comparing

$$h(w) \text{ and } h(w') .$$

EXAMPLES

Example 1 For a vector $a = (a_1, \ldots, a_k)$ of integers let

$$H(a) = \sum_{i=0}^{k} a_i \mod n$$

where n is a product of two large integers.
This hash functions does not meet any of the three properties mentioned on the last slide.

Example 1 For a vector $a = (a_1, \ldots, a_k)$ of integers let

$$H(a) = \sum_{i=0}^{k} a_i \mod n$$

where n is a product of two large integers.
This hash functions does not meet any of the three properties mentioned on the last slide.
Example 2 For a vector $a = (a_1, \ldots, a_k)$ of integers let

$$H(a) = (\sum_{i=0}^{k} a_i)^2 \mod n$$

This fuction is one-way, but it is not weakly collision-free.

Theorem Let $h : X \rightarrow Z$ be a hash function where X and Z are finite and $|X| \geq 2|Z|$. If there is an inversion algorithm **A** for h, then there exists randomized algorithm to find collisions.

FINDING COLLISIONS with INVERSION ALGORITHM

Theorem Let $h : X \rightarrow Z$ be a hash function where X and Z are finite and $|X| \geq 2|Z|$. If there is an inversion algorithm **A** for h, then there exists randomized algorithm to find collisions.

Sketch of the proof. One can easily show that the following algorithm

1. Choose a random $x \in X$ and compute z=h(x); Compute $x_1 = \mathbf{A}(z)$;
2. **if** $x_1 \neq x$, then x_1 and x collide (under h – success) **else** failure

has probability of success

$$p(success) = \frac{1}{|X|} \sum_{x \in X} \frac{|[x]| - 1}{|[x]|} \geq \frac{1}{2}$$

where, for $x \in X$, [x] is the set of elements having the same hash as x.

VARIATIONS on BIRTHDAY PARADOX

It is well known that if there are 23 (29) [40] $\{57\} < 100 >$ people in one room, then the probability that two of them have the same birthday is more than 50% (70%)[89%] $\{99\%\} < 99.99997\% >$ — this is called a Birthday paradox.

VARIATIONS on BIRTHDAY PARADOX

It is well known that if there are 23 (29) [40] $\{57\}$ $< 100 >$ people in one room, then the probability that two of them have the same birthday is more than 50% (70%)[89%] $\{99\%\}$ $< 99.99997\% >$ — this is called a Birthday paradox.

More generally, if we have n objects and r people, each choosing one object (so that several people can choose the same object), then if $r \approx 1.177\sqrt{n}(r \approx \sqrt{2n\lambda})$, then probability that two people choose the same object is 50% ($(1 - e^{-\lambda})\%$).

It is well known that if there are 23 (29) [40] $\{57\}$ $< 100 >$ people in one room, then the probability that two of them have the same birthday is more than 50% (70%)[89%] $\{99\%\}$ $< 99.99997\% >$ — this is called a Birthday paradox.

More generally, if we have n objects and r people, each choosing one object (so that several people can choose the same object), then if $r \approx 1.177\sqrt{n}(r \approx \sqrt{2n\lambda})$, then probability that two people choose the same object is 50% $((1 - e^{-\lambda})\%)$.

Another version of the birthday paradox: Let us have n objects and two groups of r people. If $r \approx \sqrt{\lambda n}$, then probability that someone from one group chooses the same object as someone from the other group is $(1 - e^{-\lambda})$.

For probability $\bar{p}(n)$ that all n people in a room have birthday in different days, it holds

$$\bar{p}(n) = \prod_{i=1}^{n-1}\left(1 - \frac{i}{365}\right) = \frac{\prod_{i=0}^{n-1}(365-i)}{365^n} = \frac{365!}{365^n(365-n)!}$$

BASIC DERIVATIONS related to BIRTHDAY PARADOX

For probability $\bar{p}(n)$ that all n people in a room have birthday in different days, it holds

$$\bar{p}(n) = \prod_{i=1}^{n-1}\left(1 - \frac{i}{365}\right) = \frac{\prod_{i=0}^{n-1}(365 - i)}{365^n} = \frac{365!}{365^n(365-n)!}$$

This equation expresses the fact for no person to share a birthday, the second person cannot have the same birthday as the first one, third person cannot have the same birthday as first two,.....

BASIC DERIVATIONS related to BIRTHDAY PARADOX

For probability $\bar{p}(n)$ that all n people in a room have birthday in different days, it holds

$$\bar{p}(n) = \prod_{i=1}^{n-1}\left(1 - \frac{i}{365}\right) = \frac{\prod_{i=0}^{n-1}(365-i)}{365^n} = \frac{365!}{365^n(365-n)!}$$

This equation expresses the fact for no person to share a birthday, the second person cannot have the same birthday as the first one, third person cannot have the same birthday as first two,.....

Probability $p(n)$ that at least two person have the same birthday is therefore

$$p(n) = 1 - \bar{p}(n)$$

This probability is larger than 0.5 first time for $n = 23$.

HASH FUNCTION DOMAIN LOWER BOUND

Birthday paradox imposes a lower bound on the sizes of message digests (fingerprints)

For example a 40-bit message would be insecure because a collision could be found with probability 0.5 with just over 20^{20} random hashes.

Minimum acceptable size of message digest seems to be 128 and therefore 160 are used in such important systems as DSS – Digital Signature Schemes (standard).

AN ALMOST GOOD HASH FUNCTION

We show an example of the hash function (so called Discrete Log Hash Function) that seems to have as the only drawback that it is too slow to be used in practice:

Let p be a large prime such that $q = \frac{(p-1)}{2}$ is also prime and let α, β be two primitive roots modulo p. Denote $a = \log_\alpha \beta$ (that is $\beta = \alpha^a$).

h will map two integers smaller than q to an integer smaller than p, for $m = x_0 + x_1 q, 0 \leq x_0, x_1 \leq q - 1$ as follows,

$$h(x_0, x_1) = h(m) = \alpha^{x_0} \beta^{x_1} \pmod p.$$

AN ALMOST GOOD HASH FUNCTION

We show an example of the hash function (so called Discrete Log Hash Function) that seems to have as the only drawback that it is too slow to be used in practice:

Let p be a large prime such that $q = \frac{(p-1)}{2}$ is also prime and let α, β be two primitive roots modulo p. Denote $a = \log_\alpha \beta$ (that is $\beta = \alpha^a$).

h will map two integers smaller than q to an integer smaller than p, for $m = x_0 + x_1 q, 0 \leq x_0, x_1 \leq q-1$ as follows,

$$h(x_0, x_1) = h(m) = \alpha^{x_0}\beta^{x_1} \pmod p.$$

To show that h is one-way and collision-free the following fact can be used:

FACT: If we know different messages m_1 and m_2 such that $h(m_1) = h(m_2)$, then we can compute $\log_\alpha \beta$.

EXTENDING HASH FUNCTIONS

Let $h : \{0,1\}^m \rightarrow \{0,1\}^t$ be a strongly collision-free hash function, where $m > t + 1$.

We design now a strongly collision-free hash function

$$h^* : \sum_{i=m}^{\infty} \{0,1\}^i \rightarrow \{0,1\}^t.$$

Let a bit string x, $|x| = n > m$, have decomposition

$$x = x_1 \| x_2 \ldots \| x_k \ ,$$

where $|x_i| = m - t - 1$ if $i < k$ and $|x_k| = m - t - 1 - d$ for some d. (Hence $k = \left\lceil \frac{n}{(m-t-1)} \right\rceil$.)

h^* will be computed as follows:

1. **for** i=1 **to** k-1 **do** $y_i := x_i$;
2. $y_k := x_k \| 0^d$; $y_{k+1} :=$ binary representation of d ;
3. $g_1 := h(0^{t+1} \| y_1)$;
4. **for** i=1 **to** k **do** $g_{i+1} := h(g_i \| 1 \| y_{i+1})$;
5. $h^*(x) := g_{k+1}$.

HASH FUNCTIONS from CRYPTOSYSTEMS

Let us have computationally secure cryptosystem with plaintexts, keys and cryptotexts being binary strings of a fixed length n and with encryption function e_k.

If

$$x = x_1 \| x_2 \| \dots \| x_k$$

is decomposition of x into substrings of length n, g_0 is a random string, and

$$g_i = f(x_i, g_{i-1})$$

for $i = 1, \dots, k$, where f is a function that "incorporates" encryption function e_k of the cryptosystem, then

$$h(x) = g_k \ .$$

For example such good properties have these two functions:

$$f(x_i, g_{i-1}) = e_{g_{i-1}}(x_i) \oplus x_i$$
$$f(x_i, g_{i-1}) = e_{g_{i-1}}(x_i) \oplus x_i \oplus g_{i-1}$$

PRACTICALLY USED HASH FUNCTIONS

A variety of hash functions has been constructed. Very often used hash functions are MD4, MD5 (created by Rivest in 1990 and 1991 and producing 128 bit message digest).

NIST even published, as a standard, in 1993, SHA (Secure Hash Algorithm) – producing 160 bit message digest – based on similar ideas as MD4 and MD5.

A hash function is called secure if it is strongly collision-free.

One of the most important cryptographic results of the last years was due to the Chinese Wang who has shown that MD4 is not cryptographically secure.

RANDOMIZED VERSION of RSA-LIKE CRYPTOSYSTEM

The scheme works for any trapdoor function (as in case of RSA),

$$f : D \rightarrow D, D \subset \{0,1\}^n,$$

for any pseudorandom generator

$$G : \{0,1\}^k \rightarrow \{0,1\}^l,\ k << l$$

and any hash function

$$h : \{0,1\}^l \rightarrow \{0,1\}^k,$$

where $n = l + k$. Given a random seed $s \in \{0,1\}^k$ as input, G generates a pseudorandom bit-sequence of length l.

RANDOMIZED VERSION of RSA-LIKE CRYPTOSYSTEM

The scheme works for any trapdoor function (as in case of RSA),

$$f : D \to D, D \subset \{0,1\}^n,$$

for any pseudorandom generator

$$G : \{0,1\}^k \to \{0,1\}^l,\ k << l$$

and any hash function

$$h : \{0,1\}^l \to \{0,1\}^k,$$

where $n = l + k$. Given a random seed $s \in \{0,1\}^k$ as input, G generates a pseudorandom bit-sequence of length l.

Encryption of a message $m \in \{0,1\}^l$ is done as follows:

1. A random string $r \in \{0,1\}^k$ is chosen.
2. Set $x = (m \oplus G(r)) \| (r \oplus h(m \oplus G(r)))$. (If $x \notin D$ go to step 1.)
3. Compute encryption $c = f(x)$ – length of x and of c is n.

RANDOMIZED VERSION of RSA-LIKE CRYPTOSYSTEM

The scheme works for any trapdoor function (as in case of RSA),

$$f : D \rightarrow D, D \subset \{0,1\}^n,$$

for any pseudorandom generator

$$G : \{0,1\}^k \rightarrow \{0,1\}^l,\ k << l$$

and any hash function

$$h : \{0,1\}^l \rightarrow \{0,1\}^k,$$

where $n = l + k$. Given a random seed $s \in \{0,1\}^k$ as input, G generates a pseudorandom bit-sequence of length l.

Encryption of a message $m \in \{0,1\}^l$ is done as follows:

1. A random string $r \in \{0,1\}^k$ is chosen.
2. Set $x = (m \oplus G(r)) \| (r \oplus h(m \oplus G(r)))$. (If $x \notin D$ go to step 1.)
3. Compute encryption c = f(x) – length of x and of c is n.

Decryption of a cryptotext c.

- Compute $f^{-1}(c) = a\|b$, $|a| = l$ and $|b| = k$.
- Set $r = h(a) \oplus b$ and get $m = a \oplus G(r)$.

Comment Operation "$\|$" stands for a concatenation of strings.

BLOOM-GOLDWASSER CRYPTOSYSTEM ONCE MORE

Private key: Blum primes p and q.

BLOOM-GOLDWASSER CRYPTOSYSTEM ONCE MORE

Private key: Blum primes p and q.

Public key: $n = pq$.

BLOOM-GOLDWASSER CRYPTOSYSTEM ONCE MORE

Private key: Blum primes p and q.
Public key: n = pq.
Encryption of $x \in \{0,1\}^m$.

1. Randomly choose $s_0 \in \{0, 1, \ldots, n\}$.
2. For I = 1, 2, ..., m + 1 compute

$$s_i \leftarrow s_{i-1}^2 \mod n$$

   and $\sigma_i = lsb(s_i)$.

The cryptotext is (s_{m+1}, y), where $y = x \oplus \sigma_1\sigma_2 \ldots \sigma_m$.

BLOOM-GOLDWASSER CRYPTOSYSTEM ONCE MORE

Private key: Blum primes p and q.

Public key: n = pq.

Encryption of $x \in \{0,1\}^m$.

1. Randomly choose $s_0 \in \{0,1,\dots,n\}$.
2. For I = 1, 2, ..., m + 1 compute

$$s_i \leftarrow s_{i-1}^2 \mod n$$

   and $\sigma_i = lsb(s_i)$.

The cryptotext is (s_{m+1}, y), where $y = x \oplus \sigma_1\sigma_2\dots\sigma_m$.

Decryption: of the cryptotext (r, y):

Let $d = 2^{-m} \bmod \phi(n))$.

- Let $s_1 = r^d \mod n$.
- For i = 1, ..., m, compute $\sigma_i = lsb(s_i)$ and $s_{i+1} \leftarrow s_i^2 \mod n$

The plaintext x can then be computed as $y \oplus \sigma_1\sigma_2\dots\sigma_m$.

APENDIX

GLOBAL GOALS of CRYPTOGRAPHY

Cryptosystems and encryption/decryption techniques are only one part of modern cryptography.

General goal of modern cryptography is construction of schemes which are robust against malicious attempts to make these schemes to deviate from their prescribed functionality.

The fact that an adversary can design its attacks after the cryptographic scheme has been specified, makes design of such cryptographic schemes very difficult – schemes should be secure under all possible attacks.

In the next chapters several of such most important basic functionalities and design of secure systems for them will be considered. For example: digital signatures, user and message authentication,. . .

Moreover, also such basic primitives as zero-knowledge proofs, needed to deal with general cryptography problems will be presented and discussed.

We will also discuss cryptographic protocols for a variety of important applications. For example for voting, digital cash,. . .

BLUM INTEGERS

- An integer n is a Blum integer if $n = pq$, where p, q are primes congruent 3 modulo 4, that is primes of the form $4k + 3$ for some integer k.
- If n is a Blum integer, then each $x \in QR(n)$ has 4 square roots and exactly one of them is in $QR(n)$ – so called principal square root of x modulo n.
- Function $f : QR(n) \rightarrow QR(n)$ defined by $f(x) = x^2 \bmod n$ is a permutation.

Part VII

Digital signatures

CHAPTER 7: DIGITAL SIGNATURES

Digital signatures are one of the most important inventions/applications of modern cryptography.

The problem is how can a user sign a message such that everybody (or the intended addressee only) can verify the digital signature and the signature is good enough also for legal purposes.

Digital signatures are one of the most important inventions/applications of modern cryptography.

The problem is how can a user sign a message such that everybody (or the intended addressee only) can verify the digital signature and the signature is good enough also for legal purposes.

Example: Assume that each user A uses a public-key cryptosystem (e_A,d_A).

A way to sign a message w by a user A, so that any user can verify the signature:

$$d_A(w)$$

CHAPTER 7: DIGITAL SIGNATURES

Digital signatures are one of the most important inventions/applications of modern cryptography.

The problem is how can a user sign a message such that everybody (or the intended addressee only) can verify the digital signature and the signature is good enough also for legal purposes.

Example: Assume that each user A uses a public-key cryptosystem (e_A,d_A).

A way to sign a message w by a user A, so that any user can verify the signature:

$$d_A(w)$$

A way to sign a message w by a user A so that only user B can verify the signature:

$$e_B(d_A(w))$$

CHAPTER 7: DIGITAL SIGNATURES

Digital signatures are one of the most important inventions/applications of modern cryptography.

The problem is how can a user sign a message such that everybody (or the intended addressee only) can verify the digital signature and the signature is good enough also for legal purposes.

Example: Assume that each user A uses a public-key cryptosystem (e_A,d_A).

A way to sign a message w by a user A, so that any user can verify the signature:

$$d_A(w)$$

A way to sign a message w by a user A so that only user B can verify the signature:

$$e_B(d_A(w))$$

Digital signatures are one of the most important inventions/applications of modern cryptography.

The problem is how can a user sign a message such that everybody (or the intended addressee only) can verify the digital signature and the signature is good enough also for legal purposes.

Example: Assume that each user A uses a public-key cryptosystem (e_A,d_A).

A way to sign a message w by a user A, so that any user can verify the signature:

$$d_A(w)$$

A way to sign a message w by a user A so that only user B can verify the signature:

$$e_B(d_A(w))$$

Example Assume Alice succeeds to factor the integer Bob used, as modulus, to sign his will, using RSA, 20 years ago. Even if the key has already expired, Alice can rewrite Bob's will, leaving fortune to her, and date it 20 years ago.

Moral: It may pay off to factor a single integers using many years of many computers power.

DIGITAL SIGNATURES - BASIC GOALS

Digital signatures should be such that each user should be able to verify signatures of other users, but that should give him/her no information how to sign a message on behalf of other users.

DIGITAL SIGNATURES - BASIC GOALS

Digital signatures should be such that each user should be able to verify signatures of other users, but that should give him/her no information how to sign a message on behalf of other users.

An important difference from a handwritten signature is that digital signature of a message is always intimately connected with the message, and for different messages is different, whereas the handwritten signature is adjoined to the message and always looks the same.

DIGITAL SIGNATURES - BASIC GOALS

Digital signatures should be such that each user should be able to verify signatures of other users, but that should give him/her no information how to sign a message on behalf of other users.

An important difference from a handwritten signature is that digital signature of a message is always intimately connected with the message, and for different messages is different, whereas the handwritten signature is adjoined to the message and always looks the same.

Technically, a digital signature signing is performed by a signing algorithm and a digital signature is verified by a verification algorithm.

A copy of a digital (classical) signature is identical (usually distinguishable) to (from) the origin. A care has therefore to be taken that digital signatures are not misused.

This chapter contains some of the main techniques for design and verification of digital signatures (as well as some possible attacks on them).

Can we make digital signatures by digitalizing our usual signature and attaching them to the messages (documents) that need to be signed?

No, because such signatures could be easily removed and attached to some other documents or messages.

Key observation: Digital signatures have to depend not only on the signer, but also on the message that is being signed.

A SCHEME of DIGITAL SIGNATURE SYSTEMS – SIMPLIFIED VERSION

A **digital signature system** (**DSS**) consists of:

- P - the space of possible plaintexts (messages).
- S - the space of possible signatures.
- K - the space of possible keys.

A SCHEME of DIGITAL SIGNATURE SYSTEMS – SIMPLIFIED VERSION

A **digital signature system** (**DSS**) consists of:

- P - the space of possible plaintexts (messages).
- S - the space of possible signatures.
- K - the space of possible keys.
- For each $k \in K$ there is a signing algorithm sig_k and a corresponding verification algorithm ver_k such that

$$sig_k : P \rightarrow S.$$

$$ver_k : P \otimes S \rightarrow \{true, false\}$$

and

$$ver_k(w, s) = \begin{cases} true & \text{if } \; s = sig_k(w);, \\ false & \text{otherwise.} \end{cases}$$

Algorithms sig_k and ver_k should be computable in polynomial time.

A SCHEME of DIGITAL SIGNATURE SYSTEMS – SIMPLIFIED VERSION

A **digital signature system** (**DSS**) consists of:

- P - the space of possible plaintexts (messages).
- S - the space of possible signatures.
- K - the space of possible keys.
- For each $k \in K$ there is a signing algorithm sig_k and a corresponding verification algorithm ver_k such that

$$sig_k : P \rightarrow S.$$

$$ver_k : P \otimes S \rightarrow \{true, false\}$$

  and

$$ver_k(w, s) = \begin{cases} true & \text{if } \; s = sig_k(w);, \\ false & \text{otherwise.} \end{cases}$$

Algorithms sig_k and ver_k should be computable in polynomial time.

Verification algorithm can be publicly known; signing algorithm (actually only its key) should be kept secret

DIGITAL SIGNATURE SCHEMES I

Digital signature schemes are basic tools for authentication and non-repudiation of messages. A digital signature scheme allows anyone to verify signature of any sender S without providing any information how to generate signatures of S.

DIGITAL SIGNATURE SCHEMES I

Digital signature schemes are basic tools for authentication and non-repudiation of messages. A digital signature scheme allows anyone to verify signature of any sender S without providing any information how to generate signatures of S.

A Digital Signature Scheme (M, S, K_s, K_v) is given by:

- M - a set of messages to be signed
- S - a set of possible signatures
- K_s - a set of private keys for signing
- K_v - a set of public keys for verification

DIGITAL SIGNATURE SCHEMES I

Digital signature schemes are basic tools for authentication and non-repudiation of messages. A digital signature scheme allows anyone to verify signature of any sender S without providing any information how to generate signatures of S.

A Digital Signature Scheme (M, S, K_s, K_v) is given by:

- M - a set of messages to be signed
- S - a set of possible signatures
- K_s - a set of private keys for signing
- K_v - a set of public keys for verification

Moreover, it is required that:

- For each k from K_s, there exists a single and easy to compute signing mapping

$$sig_k\colon \{0,1\}^* \times M \rightarrow S$$

- For each k from K_v there exists a single and easy to compute verification mapping

$$ver_k\colon M \times S \rightarrow \{true,\ false\}$$

such that the following two conditions are satisfied:

DIGITAL SIGNATURES SCHEMES II

Correctness:

For each message m from M and public key k in K_v, it holds

$$ver_k(m, s) = \text{true}$$

if there is an r from $\{0, 1\}^*$ such that

$$\text{s} = sig_l(\text{r, m})$$

for a private key l from K_s corresponding to the public key k.

Security:

For any w from M and k in K_v , it is computationally infeasible, without the knowledge of the private key corresponding to k, to find a signature s from S such that

$$ver_k(\text{w, s}) = \text{true}.$$

A COMMENT ON DIGITAL SIGNATURE SCHEMES

Sometimes it is said that a digital signature scheme contains also a key generation algorithm that selects uniformly and randomly a secret key (from a set of potential secret keys) and outputs this secret key and the corresponding private key.

Basic attack models

KEY-ONLY ATTACK : The attacker is only given the public verification key.

KNOWN SIGNATURES ATTACK : The attacker is given valid signatures for several messages known but not chosen by the attacker.

CHOSEN SIGNATURES ATTACK : The attacker is given valid signatures for sever al messages chosen by the attacker.

Total break of a signature scheme: The adversary manages to recover the secret key from the public key.

BASIC ATTACKS on DIGITAL SIGNATURES

Total break of a signature scheme: The adversary manages to recover the secret key from the public key.

Universal forgery: The adversary can derive from the public key an algorithm which allows to forge the signature of any message.

BASIC ATTACKS on DIGITAL SIGNATURES

Total break of a signature scheme: The adversary manages to recover the secret key from the public key.

Universal forgery: The adversary can derive from the public key an algorithm which allows to forge the signature of any message.

Selective forgery: The adversary can derive from the public key a method to forge signatures of selected messages (where selection was made prior the knowledge of the public key).

BASIC ATTACKS on DIGITAL SIGNATURES

Total break of a signature scheme: The adversary manages to recover the secret key from the public key.

Universal forgery: The adversary can derive from the public key an algorithm which allows to forge the signature of any message.

Selective forgery: The adversary can derive from the public key a method to forge signatures of selected messages (where selection was made prior the knowledge of the public key).

Existential forgery: The adversary is able to create from the public key a valid signature of a message m (but has no control for which m).

A DIGITAL SIGNATURE of one BIT

Let us start with a very simple but much illustrating (though non-practical) example how to sign a single bit.

A DIGITAL SIGNATURE of one BIT

Let us start with a very simple but much illustrating (though non-practical) example how to sign a single bit.

Design of the signature scheme:

A one-way function f(x) is chosen.

Two integers k_0 and k_1 are chosen and kept secret by the signer, and three items

$$f,\ (0,\ s_0),\ (1,\ s_1)$$

are made public, where

$$s_0 = f(k_0),\ s_1 = f(k_1)$$

A DIGITAL SIGNATURE of one BIT

Let us start with a very simple but much illustrating (though non-practical) example how to sign a single bit.

Design of the signature scheme:

A one-way function f(x) is chosen.

Two integers k_0 and k_1 are chosen and kept secret by the signer, and three items

$$f, (0, s_0), (1, s_1)$$

are made public, where

$$s_0 = f(k_0), \ s_1 = f(k_1)$$

Signature of a bit b:

$$(b, k_b).$$

A DIGITAL SIGNATURE of one BIT

Let us start with a very simple but much illustrating (though non-practical) example how to sign a single bit.

Design of the signature scheme:

A one-way function f(x) is chosen.

Two integers k_0 and k_1 are chosen and kept secret by the signer, and three items

$$f, (0, s_0), (1, s_1)$$

are made public, where

$$s_0 = f(k_0), \; s_1 = f(k_1)$$

Signature of a bit b:

$$(b, k_b).$$

Verification of such a signature

$$s_b = f(k_b)$$

SECURITY?

RSA SIGNATURES and ATTACKS on them

Let us have an RSA cryptosystem with encryption and decryption exponents e and d and modulus n.

Signing of a message w:

$$s = (w, \sigma), \text{ where } \sigma = w^d \bmod n$$

Verification of a signature $s = (w, \sigma)$:

$$w = \sigma^e \bmod n?$$

RSA SIGNATURES and ATTACKS on them

Let us have an RSA cryptosystem with encryption and decryption exponents e and d and modulus n.

Signing of a message w:

$$s = (w, \sigma), \text{ where } \sigma = w^d \bmod n$$

Verification of a signature $s = (w, \sigma)$:

$$w = \sigma^e \bmod n?$$

Attacks

- It might happen that Bob accepts a signature not produced by Alice. Indeed, let Eve, using Alice's public key, compute w^e and say that (w^e, w) is a message signed by Alice.

  Everybody verifying Alice's signature gets $w^e = w^e$.

RSA SIGNATURES and ATTACKS on them

Let us have an RSA cryptosystem with encryption and decryption exponents e and d and modulus n.

Signing of a message w:

$$s = (w, \sigma), \text{ where } \sigma = w^d \bmod \text{n}$$

Verification of a signature $s = (w, \sigma)$:

$$w = \sigma^e \bmod \text{n?}$$

Attacks

- It might happen that Bob accepts a signature not produced by Alice. Indeed, let Eve, using Alice's public key, compute w^e and say that (w^e, w) is a message signed by Alice.

  Everybody verifying Alice's signature gets $w^e = w^e$.

- Some new signatures can be produced without knowing the secret key.

  Indeed, is σ_1 and σ_2 are signatures for w_1 and w_2, then $\sigma_1\sigma_2$ and σ_1^{-1} are signatures for $w_1 w_2$ and w_1^{-1}.

ENCRYPTIONS versus SIGNATURES

Let each user U use a cryptosystem with encryption and decryption algorithms: e_U, d_U

Let w be a message

PUBLIC-KEY ENCRYPTIONS

Encryption:	$e_U(w)$
Decryption:	$d_U\ (e_U(w))$

ENCRYPTIONS versus SIGNATURES

Let each user U use a cryptosystem with encryption and decryption algorithms: e_U, d_U

Let w be a message

PUBLIC-KEY ENCRYPTIONS

Encryption:	$e_U(w)$
Decryption:	$d_U\ (e_U(w))$

PUBLIC-KEY SIGNATURES

Signing:	$d_U(w)$
Verification of the signature:	$e_U\ (d_U(w))$

FROM PKC to DSS - again

Any public-key cryptosystem in which the plaintext and cryptotext space are the same, can be used for digital signature.

Signing of a message w by a user A so that any user can verify the signature:

$$d_A(w).$$

FROM PKC to DSS - again

Any public-key cryptosystem in which the plaintext and cryptotext space are the same, can be used for digital signature.

Signing of a message w by a user A so that any user can verify the signature:

$$d_A(w).$$

Signing of a message w by a user A so that only user B can verify the signature:

$$e_B(d_A(w)).$$

FROM PKC to DSS - again

Any public-key cryptosystem in which the plaintext and cryptotext space are the same, can be used for digital signature.

Signing of a message w by a user A so that any user can verify the signature:

$$d_A(w).$$

Signing of a message w by a user A so that only user B can verify the signature:

$$e_B(d_A(w)).$$

Sending a message w and a signed message digest of w obtained by using a (standard) hash function h:

$$(w, d_A(h(w))).$$

FROM PKC to DSS - again

Any public-key cryptosystem in which the plaintext and cryptotext space are the same, can be used for digital signature.

Signing of a message w by a user A so that any user can verify the signature:

$$d_A(w).$$

Signing of a message w by a user A so that only user B can verify the signature:

$$e_B(d_A(w)).$$

Sending a message w and a signed message digest of w obtained by using a (standard) hash function h:

$$(w, d_A(h(w))).$$

If only signature (but not the encryption of the message) are of importance, then it suffices that Alice sends to Bob

$$(w, d_A(w)).$$

ElGamal SIGNATURES

Design of the ElGamal digital signature system: choose: prime p, integers $1 \leq q \leq x \leq p$, where q is a primitive element of Z_p^*;

$$\text{Compute: } y = q^x \mod p$$

$$\textbf{key K} = (\mathrm{p, q, x, y})$$

public key (p, q, y) - trapdoor: x

ElGamal SIGNATURES

Design of the ElGamal digital signature system: choose: prime p, integers $1 \leq q \leq x \leq p$, where q is a primitive element of Z_p^*;

$$\text{Compute: } y = q^x \mod p$$

$$\textbf{key K} = (\text{p, q, x, y})$$

public key (p, q, y) - trapdoor: x

Signature of a message w: Let $r \in Z_{p-1}^*$ be randomly chosen and kept secret.

$$\text{sig(w, r)} = \text{(a, b)},$$

$$\text{where } a = q^r \mod p$$

$$\text{and b} = (w - xa)r^{-1} \pmod{(p-1)}.$$

ElGamal SIGNATURES

Design of the ElGamal digital signature system: choose: prime p, integers $1 \leq q \leq x \leq p$, where q is a primitive element of Z_p^*;

$$\text{Compute: } y = q^x \mod p$$

$$\textbf{key K} = (\text{p, q, x, y})$$

public key (p, q, y) - trapdoor: x

Signature of a message w: Let $r \in Z_{p-1}^*$ be randomly chosen and kept secret.

$$\text{sig(w, r)} = \text{(a, b)},$$

$$\text{where } a = q^r \mod p$$

$$\text{and b} = (w - xa)r^{-1} \pmod{(p-1)}.$$

Verification: accept a signature (a,b) of w as valid if

$$y^a a^b \equiv q^w \pmod p$$

$$(\text{Indeed: } y^a a^b \equiv q^{ax} q^{rb} \equiv q^{ax+w-ax+k(p-1)} \equiv q^w \pmod p)$$

ElGamal SIGNATURE - EXAMPLE

Example

choose: p = 11, q = 2, x = 8

compute: $y = 2^8 \bmod 11 = 3$

w = 5 is signed as (a,b), where $a = q^r \bmod p, w = xa + rb \bmod (p-1)$

choose r = 9 – (this choice is O.K. because gcd(9, 10) = 1)

compute a $= 2^9 \bmod 11 = 6$

solve equation: $5 \equiv 8 \cdot 6 + 9b \pmod{10}$

that is $7 \equiv 9b \pmod{10} \Rightarrow b=3$

signature: (6, 3)

SECURITY of ElGamal SIGNATURES

Let us analyze several ways an eavesdropper Eve can try to forge ElGamal signature (with x - secret; p, q and y = q^x mod p - public):

$$\text{sig}(w, r) = (a, b);$$

where r is random and a = q^r mod p; b = $(w - xa)r^{-1} \pmod{p-1}$.

1. First suppose Eve tries to forge signature for a new message w, without knowing x.
   - If Eve first chooses a value a and tries to find the corresponding b, it has to compute the discrete logarithm

     $$lg_a q^w y^{-a},$$

     (because $a^b \equiv q^{r(w-xa)r^{-1}} \equiv q^{w-xa} \equiv q^w y^{-a}$) what is infeasible.
   - If Eve first chooses b and then tries to find a, she has to solve the equation

     $$y^a a^b \equiv q^{xa} q^{rb} \equiv q^w \pmod{p}.$$

     It is not known whether this equation can be solved for any given b efficiently.

SECURITY of ElGamal SIGNATURES

Let us analyze several ways an eavesdropper Eve can try to forge ElGamal signature (with x - secret; p, q and y = q^x mod p - public):

$$\text{sig}(w, r) = (a, b);$$

where r is random and a = q^r mod p; b = $(w - xa)r^{-1} \pmod{p-1}$.

1. First suppose Eve tries to forge signature for a new message w, without knowing x.
   - If Eve first chooses a value a and tries to find the corresponding b, it has to compute the discrete logarithm

     $$lg_a q^w y^{-a},$$

     (because $a^b \equiv q^{r(w-xa)r^{-1}} \equiv q^{w-xa} \equiv q^w y^{-a}$) what is infeasible.
   - If Eve first chooses b and then tries to find a, she has to solve the equation

     $$y^a a^b \equiv q^{xa} q^{rb} \equiv q^w \pmod{p}.$$

     It is not known whether this equation can be solved for any given b efficiently.
2. If Eve chooses a and b and tries to determine such w that (a,b) is signature of w, then she has to compute discrete logarithm

   $$lg_q y^a a^b.$$

   Hence, Eve can not sign a “random” message this way.

FORGING and MISUSING of ElGamal SIGNATURES

There are ways to produce, using ElGamal signature scheme, some valid forged signatures, but they do not allow an opponent to forge signatures on messages of his/her choice.

For example, if $0 \leq i, j \leq p-2$ and gcd(j, p - 1) = 1, then for

$$a = q^i y^j \mod p;\ b = -aj^{-1} \mod (p-1);\ w = -aij^{-1} \mod (p-1)$$

the pair

(a, b) is a valid signature of the message w.

This can be easily shown by checking the verification condition.

There are several ways ElGamal signatures can be broken if they are not used carefully enough.

For example, the random r used in the signature should be kept secret. Otherwise the system can be broken and signatures forged. Indeed, if r is known, then x can be computed by

$$x = (w - rb)a^{-1} \mod (p-1)$$

and once x is known Eve can forge signatures at will.

Another misuse of the ElGamal signature system is to use the same r to sign two messages. In such a case x can be computed and the system can be broken.

From ElGamal to DSA (DIGITAL SIGNATURE STANDARD)

DSA, accepted in 1994, is a modification of ElGamal digital signature scheme. It was proposed in August 1991 and adopted in December 1994.

From ElGamal to DSA (DIGITAL SIGNATURE STANDARD)

DSA, accepted in 1994, is a modification of ElGamal digital signature scheme. It was proposed in August 1991 and adopted in December 1994.

Any proposal for digital signature standard has to go through a very careful scrutiny. Why?

From ElGamal to DSA (DIGITAL SIGNATURE STANDARD)

DSA, accepted in 1994, is a modification of ElGamal digital signature scheme. It was proposed in August 1991 and adopted in December 1994.

Any proposal for digital signature standard has to go through a very careful scrutiny. Why?

Encryption of a message is usually done only once and therefore it usually suffices to use a cryptosystem that is secure **at the time of the encryption.**

On the other hand, a signed message could be a contract or a will and it can happen that it will be needed to verify a signature **many years after the message is signed.**

From ElGamal to DSA (DIGITAL SIGNATURE STANDARD)

DSA, accepted in 1994, is a modification of ElGamal digital signature scheme. It was proposed in August 1991 and adopted in December 1994.

Any proposal for digital signature standard has to go through a very careful scrutiny. Why?

Encryption of a message is usually done only once and therefore it usually suffices to use a cryptosystem that is secure **at the time of the encryption.**

On the other hand, a signed message could be a contract or a will and it can happen that it will be needed to verify a signature **many years after the message is signed.**

Since ElGamal signature is no more secure than discrete logarithm, it is necessary to use large p, with at least 512 bits.

However, with ElGamal this would lead to signatures with at least 1024 bits what is too much for such applications as smart cards.

DIGITAL SIGNATURE STANDARD I

In December 1994, on the proposal of the National Institute of Standards and Technology, the following Digital Signature Algorithm (DSA) was accepted as a standard.

DIGITAL SIGNATURE STANDARD I

In December 1994, on the proposal of the National Institute of Standards and Technology, the following Digital Signature Algorithm (DSA) was accepted as a standard.

Design of DSA

1. The following global public key components are chosen:
   - p - a random l-bit prime, $512 \leq l \leq 1024$, l = 64k.
   - q - a random 160-bit prime dividing p -1.
   - r = $h^{(p-1)/q} \mod p$, where h is a random primitive element of Z_p, such that $r > 1$, $r \neq 1$ (observe that r is a q-th root of 1 mod p).

DIGITAL SIGNATURE STANDARD I

In December 1994, on the proposal of the National Institute of Standards and Technology, the following Digital Signature Algorithm (DSA) was accepted as a standard.

Design of DSA

1. The following global public key components are chosen:
   - p - a random l-bit prime, $512 \leq l \leq 1024$, l = 64k.
   - q - a random 160-bit prime dividing p -1.
   - r = $h^{(p-1)/q} \mod p$, where h is a random primitive element of Z_p, such that $r > 1$, $r \neq 1$ (observe that r is a q-th root of 1 mod p).
2. The following user's private key component is chosen:
   - x - a random integer (once), $0 < x < q$,
3. The following value is also made public
   - y = r^x mod p.

DIGITAL SIGNATURE STANDARD I

In December 1994, on the proposal of the National Institute of Standards and Technology, the following Digital Signature Algorithm (DSA) was accepted as a standard.

Design of DSA

1. The following global public key components are chosen:
   - p - a random l-bit prime, $512 \leq l \leq 1024$, l = 64k.
   - q - a random 160-bit prime dividing p -1.
   - r = $h^{(p-1)/q} \mod p$, where h is a random primitive element of Z_p, such that $r > 1$, $r \neq 1$ (observe that r is a q-th root of 1 mod p).
2. The following user's private key component is chosen:
   - x - a random integer (once), $0 < x < q$,
3. The following value is also made public
   - y = r^x mod p.
4. Key is K = (p, q, r, x, y)

Signing and Verification

Signing of a 160-bit plaintext w

- choose random $0 < k < q$
- compute a $= (r^k \mod p) \bmod \mathrm{q}$
- compute b $= k^{-1}(\mathrm{w} + \mathrm{xa}) \bmod \mathrm{q}$ where $kk^{-1} \equiv 1 \pmod{q}$
- signature: sig(w, k) = (a, b)

Signing and Verification

Signing of a 160-bit plaintext w

- choose random $0 < k < q$
- compute a $= (r^k \mod p) \bmod q$
- compute b $= k^{-1}(w + xa) \bmod q$ where $kk^{-1} \equiv 1 \pmod{q}$
- signature: sig(w, k) = (a, b)

Verification of signature (a, b)

- compute $z = b^{-1} \mod q$
- compute $u_1 = wz \mod q$, $u_2 = az \mod q$

  verification:

$$ver_K(w, a, b) = true \Leftrightarrow (r^{u_1} y^{u_2} \mod p) \mod q = a$$

From ElGamal to DSA - II

DSA is a modification of ElGamal digital signature scheme. It was proposed in August 1991 and adopted in December 1994.

From ElGamal to DSA - II

DSA is a modification of ElGamal digital signature scheme. It was proposed in August 1991 and adopted in December 1994.

Since ElGamal signature is no more secure than discrete logarithm, it is necessary to use large p, with at least 512 bits.

However, with ElGamal this would lead to signatures with at least 1024 bits what is too much for such applications as smart cards.

In DSA a 160 bit message is signed using 320-bit signature, but computation is done modulo with 512-1024 bits.

From ElGamal to DSA - II

DSA is a modification of ElGamal digital signature scheme. It was proposed in August 1991 and adopted in December 1994.

Since ElGamal signature is no more secure than discrete logarithm, it is necessary to use large p, with at least 512 bits.

However, with ElGamal this would lead to signatures with at least 1024 bits what is too much for such applications as smart cards.

In DSA a 160 bit message is signed using 320-bit signature, but computation is done modulo with 512-1024 bits.

Observe that y and a are also q-roots of 1. Hence any exponents of r, y and a can be reduced modulo q without affecting the verification condition.

Fiat-Shamir SIGNATURE SCHEME

Choose primes p, q, compute n = pq and choose: as a public key integers $v_1, \ldots, v_k$ and compute, as a secret key, $s_1, \ldots, s_k, s_i = \sqrt{v_i^{-1}} \mod n$.

Fiat-Shamir SIGNATURE SCHEME

Choose primes p, q, compute n = pq and choose: as a public key integers $v_1, \ldots, v_k$ and compute, as a secret key, $s_1, \ldots, s_k, s_i = \sqrt{v_i^{-1}} \mod n$.

Protocol for Alice to sign a message w:

1. Alice chooses (as a security parameter) an integer t, t random integers $1 \leq r_1, \ldots, r_t < n$, and computes $x_i = r_i^2 \mod n, 1 \leq i \leq t$.

Fiat-Shamir SIGNATURE SCHEME

Choose primes p, q, compute n = pq and choose: as a public key integers $v_1, \ldots, v_k$ and compute, as a secret key, $s_1, \ldots, s_k, s_i = \sqrt{v_i^{-1}} \mod n$.

Protocol for Alice to sign a message w:

1. Alice chooses (as a security parameter) an integer t, t random integers $1 \leq r_1, \ldots, r_t < n$, and computes $x_i = r_i^2 \mod n, 1 \leq i \leq t$.
2. Alice uses a publicly known hash function h to compute $H = h(wx_1x_2 \ldots x_t)$ and then uses the first kt bits of H, denoted as b_{ij}, $1 \leq i \leq t, 1 \leq j \leq k$ as follows.

Fiat-Shamir SIGNATURE SCHEME

Choose primes p, q, compute n = pq and choose: as a public key integers $v_1, \ldots, v_k$ and compute, as a secret key, $s_1, \ldots, s_k, s_i = \sqrt{v_i^{-1}} \mod n$.

Protocol for Alice to sign a message w:

1. Alice chooses (as a security parameter) an integer t, t random integers $1 \leq r_1, \ldots, r_t < n$, and computes $x_i = r_i^2 \mod n, 1 \leq i \leq t$.
2. Alice uses a publicly known hash function h to compute $H = h(wx_1x_2 \ldots x_t)$ and then uses the first kt bits of H, denoted as b_{ij}, $1 \leq i \leq t, 1 \leq j \leq k$ as follows.
3. Alice computes $y_1, \ldots, y_t$

$$y_i = r_i \prod_{j=1}^{k} s_j^{b_{ij}} \mod n$$

Fiat-Shamir SIGNATURE SCHEME

Choose primes p, q, compute n = pq and choose: as a public key integers $v_1, \dots, v_k$ and compute, as a secret key, $s_1, \dots, s_k, s_i = \sqrt{v_i^{-1}} \mod n$.

Protocol for Alice to sign a message w:

1. Alice chooses (as a security parameter) an integer t, t random integers $1 \leq r_1, \dots, r_t < n$, and computes $x_i = r_i^2 \mod n, 1 \leq i \leq t$.
2. Alice uses a publicly known hash function h to compute $H = h(wx_1x_2 \dots x_t)$ and then uses the first kt bits of H, denoted as b_{ij}, $1 \leq i \leq t, 1 \leq j \leq k$ as follows.
3. Alice computes $y_1, \dots, y_t$
$$y_i = r_i \prod_{j=1}^{k} s_j^{b_{ij}} \mod n$$
4. Alice sends to Bob w, all b_{ij}, all y_i and also h $\{$Bob already knows Alice's public key $v_1, \dots, v_k\}$

Fiat-Shamir SIGNATURE SCHEME

Choose primes p, q, compute n = pq and choose: as a public key integers $v_1, \ldots, v_k$ and compute, as a secret key, $s_1, \ldots, s_k, s_i = \sqrt{v_i^{-1}} \mod n$.

Protocol for Alice to sign a message w:

1. Alice chooses (as a security parameter) an integer t, t random integers $1 \leq r_1, \ldots, r_t < n$, and computes $x_i = r_i^2 \mod n, 1 \leq i \leq t$.
2. Alice uses a publicly known hash function h to compute $H = h(wx_1x_2 \ldots x_t)$ and then uses the first kt bits of H, denoted as b_{ij}, $1 \leq i \leq t, 1 \leq j \leq k$ as follows.
3. Alice computes $y_1, \ldots, y_t$
$$y_i = r_i \prod_{j=1}^{k} s_j^{b_{ij}} \mod n$$
4. Alice sends to Bob w, all b_{ij}, all y_i and also h $\{$Bob already knows Alice's public key $v_1, \ldots, v_k\}$
5. Bob computes $z_1, \ldots, z_k$
$$Z_i = y_i^2 \prod_{j=1}^{k} v_j^{b_{ij}} \mod n = r_i^2 \prod_{j=1}^{k} (v_j^{-1})^{b_{ij}} \prod_{j=1}^{k} v_j^{b_{ij}} = r_i^2 = x_i$$
and verifies that the first $k \times t$ bits of $h(wx_1x_2 \ldots x_t)$ are the b_{ij} values that Alice has sent to him.

Fiat-Shamir SIGNATURE SCHEME

Choose primes p, q, compute n = pq and choose: as a public key integers $v_1, \ldots, v_k$ and compute, as a secret key, $s_1, \ldots, s_k, s_i = \sqrt{v_i^{-1}} \mod n$.

Protocol for Alice to sign a message w:

1. Alice chooses (as a security parameter) an integer t, t random integers $1 \leq r_1, \ldots, r_t < n$, and computes $x_i = r_i^2 \mod n, 1 \leq i \leq t$.
2. Alice uses a publicly known hash function h to compute $H = h(wx_1x_2 \ldots x_t)$ and then uses the first kt bits of H, denoted as b_{ij}, $1 \leq i \leq t, 1 \leq j \leq k$ as follows.
3. Alice computes $y_1, \ldots, y_t$
$$y_i = r_i \prod_{j=1}^{k} s_j^{b_{ij}} \mod n$$
4. Alice sends to Bob w, all b_{ij}, all y_i and also h {Bob already knows Alice's public key $v_1, \ldots, v_k$}
5. Bob computes $z_1, \ldots, z_k$
$$Z_i = y_i^2 \prod_{j=1}^{k} v_j^{b_{ij}} \mod n = r_i^2 \prod_{j=1}^{k} (v_j^{-1})^{b_{ij}} \prod_{j=1}^{k} v_j^{b_{ij}} = r_i^2 = x_i$$
and verifies that the first $k \times t$ bits of $h(wx_1x_2 \ldots x_t)$ are the b_{ij} values that Alice has sent to him.

   Security of this signature scheme is 2^{-kt}.

   Advantage over the RSA-based signature scheme: only about 5% of modular multiplications are needed.

SAD STORY

SAD STORY

Alice and Bob got to jail - and, unfortunately, to different jails.

SAD STORY

Alice and Bob got to jail - and, unfortunately, to different jails.

Walter, the warden, allows them to communicate by network, but he will not allow their messages to be encrypted.

SAD STORY

Alice and Bob got to jail - and, unfortunately, to different jails.

Walter, the warden, allows them to communicate by network, but he will not allow their messages to be encrypted.

Problem: Can Alice and Bob set up a subliminal channel, a covert communication channel between them, in full view of Walter, even though the messages themselves that they exchange contain no secret information?

Ong-Schnorr-Shamir SUBLUMINAL CHANNEL SCHEME

Story Alice and Bob are in different jails. Walter, the warden, allows them to communicate by network, but he will not allow messages to be encrypted. Can they set up a subliminal channel, a covert communication channel between them, in full view of Walter, even though the messages themselves contain no secret information?

Ong-Schnorr-Shamir SUBLUMINAL CHANNEL SCHEME

Story Alice and Bob are in different jails. Walter, the warden, allows them to communicate by network, but he will not allow messages to be encrypted. Can they set up a subliminal channel, a covert communication channel between them, in full view of Walter, even though the messages themselves contain no secret information?

Yes. Alice and Bob create first the following communication scheme:

They choose a large n and an integer k such that gcd(n, k) = 1.

They calculate $h = k^{-2} \mod n = (k^{-1})^2 \mod n$.

Public key: h, n

Trapdoor information: k

Let secret message Alice wants to send be w (it has to be such that gcd(w, n) =1)
Denote a harmless message she uses by w' (it has to be such that gcd(w ',n) = 1)
Signing by Alice:

$$S_1 = \frac{1}{2} \cdot (\frac{w'}{w} + w) \mod n$$
$$S_2 = \frac{k}{2} \cdot (\frac{w'}{w} - w) \mod n$$

Signature: (S_1, S_2). Alice then sends to Bob (w', S_1, S_2)
Signature verification method for Walter: w' $= S_1^2 - hS_2^2 (\mod n)$

Decryption by Bob: $w = \dfrac{w'}{(S_1 + k^{-1}S_2)} \mod n$

ONE-TIME SIGNATURES

Lamport signature scheme shows how to construct a signature scheme for one use only - from any one-way function.

Let k be a positive integer and let $P = \{0,1\}^k$ be the set of messages.
Let f: $Y \rightarrow Z$ be a one-way function where Y is a set of "signatures".

For $1 \leq i \leq k$, j = 0,1 let $y_{ij} \in Y$ be chosen randomly and $z_{ij} = f(y_{ij})$.

The key K consists of 2k y's and z's. y's are secret, z's are public.

ONE-TIME SIGNATURES

Lamport signature scheme shows how to construct a signature scheme for one use only - from any one-way function.

Let k be a positive integer and let $P = \{0,1\}^k$ be the set of messages.
Let f: $Y \rightarrow Z$ be a one-way function where Y is a set of "signatures".

For $1 \leq i \leq k$, j = 0,1 let $y_{ij} \in Y$ be chosen randomly and $z_{ij} = f(y_{ij})$.

The key K consists of 2k y's and z's. y's are secret, z's are public.

Signing of a message $x = x_1 \dots x_k \in \{0,1\}^k$

$$sig(x_1 \dots x_k) = (y_{1,x1}, \dots, y_{k,xk}) = (a_1, \dots, a_k) \text{ - notation}$$

and

$$ver_K(x_1 \dots x_k, a_1, \dots, a_k) = true \Leftrightarrow f(a_i) = z_{i,xi}, 1 \leq i \leq k$$

Eve cannot forge a signature because she is unable to invert one-way functions.

Important note: Lamport signature scheme can be used to sign only one message.

SIGNING of FINGERPRINTS

Signature schemes presented so far allow to sign only "short" messages. For example, DSS is used to sign 160 bit messages (with 320-bit signatures).

A naive solution is to break long message into a sequence of short ones and to sign each block separately.

Disadvantages: signing is slow and for long signatures integrity is not protected.

The solution is to use a fast public **hash function h** which maps a message of any length to a fixed length hash. The hash is then signed.

Example:

message	w	arbitrary length
message digest	z = h (w)	160bits
El Gamal signature	y = sig(z)	320bits

If Bob wants to send a signed message w he sends (w, sig(h(w))).

TIMESTAMPING

There are various ways that a digital signature can be compromised.

For example: if Eve determines the secret key of Bob, then she can forge signatures of any Bob's message she likes. If this happens, authenticity of all messages signed by Bob before Eve got the secret key is to be questioned.

The key problem is that there is no way to determine when a message was signed.

A timestamping protocol should provide a proof that a message was signed at a certain time.

TIMESTAMPING

There are various ways that a digital signature can be compromised.

For example: if Eve determines the secret key of Bob, then she can forge signatures of any Bob's message she likes. If this happens, authenticity of all messages signed by Bob before Eve got the secret key is to be questioned.

The key problem is that there is no way to determine when a message was signed.

A timestamping protocol should provide a proof that a message was signed at a certain time.

In the following pub denotes some publicly known information that could not be predicted before the day of the signature (for example, stock-market data).

TIMESTAMPING

There are various ways that a digital signature can be compromised.

For example: if Eve determines the secret key of Bob, then she can forge signatures of any Bob's message she likes. If this happens, authenticity of all messages signed by Bob before Eve got the secret key is to be questioned.

The key problem is that there is no way to determine when a message was signed.

A timestamping protocol should provide a proof that a message was signed at a certain time.

In the following pub denotes some publicly known information that could not be predicted before the day of the signature (for example, stock-market data).

Timestamping by Bob of a signature on a message w, using a hash function h.

- Bob computes $z = h(w)$;
- Bob computes $z' = h(z \parallel pub)$; – $\{ \parallel \}$ denotes concatenation
- Bob computes $y = sig(z')$;
- Bob publishes (z, pub, y) in the next days's newspaper.

It is now clear that signature could not be done after the triple (z, pub, y) was published, but also not before the date pub was known.

BLIND SIGNATURES

The basic idea is that Sender makes Signer to sign a message m without Signer knowing m, therefore blindly – this is needed in e-commerce.

Blind signing can be realized by a two party protocol, between the Sender and the Signer, that has the following properties.

- In order to sign (by a Signer) a message m, the Sender creates, using a blinding procedure, from the message m a new message $m*$ from which m can not be obtained without knowing a secret, and sends $m*$ to the Signer.
- The Signer signs the message $m*$ to get a signature s_{m^*} (of $m*$) and sends s_{m*} to the Sender. The signing is to be done in such a way that the Sender can afterwards compute, using an unblinding procedure, from Signer's signature s_{m*} of $m*$ – the signer signature s_m of m.

Chaum's BLIND SIGNATURE SCHEME

This blind signature protocol combines RSA with blinding/unblinding features.

Bob's RSA public key is (n, e) and his private key is d.
Let m be a message, $0 < m < n$,

PROTOCOL:

- Alice chooses a random $0 < k < n$ with $\gcd(n, k) = 1$.
- Alice computes $m^* = mk^e$ (**mod** n) and sends it to Bob (this way Alice blinds the message m).
- Bob computed $s^* = (m^*)^d$ (**mod** n) and sends s* to Alice (this way Bob signs the blinded message m*).
- Alice computes $s = k^{-1}s^*$(**mod** n) to obtain Bob's signature m^d of m (Alice performs unblinding of $m*$).

Verification is equivalent to that of the RSA signature scheme.

FAIL-THEN-STOP SIGNATURES

They are signatures schemes that use a trusted authority and provide ways to prove, if it is the case, that a powerful enough adversary is around who could break the signature scheme and therefore its use should be stopped.

The scheme is maintained by a trusted authority that chooses a secret key for each signer, keeps them secret, even from the signers themselves, and announces only the related public keys.

An important idea is that signing and verification algorithms are enhanced by a so-called proof-of-forgery algorithm. When the signer sees a forged signature he is able to compute his secret key and by submitting it to the trusted authority to prove the existence of a forgery and this way to achieve that any further use of the signature scheme is stopped.

So called Heyst-Pedersen Scheme is an example of a Fail-Then-Stop signature Scheme.

DIGITAL SIGNATURES with ENCRYPTION and RESENDING

1. Alice signs the message: $s_A(w)$.

DIGITAL SIGNATURES with ENCRYPTION and RESENDING

1. Alice signs the message: $s_A(w)$.
2. Alice encrypts the signed message: $e_B(s_A(w))$.
3. Bob decrypts the signed message: $d_B(e_B(s_A(w))) = s_A(w)$.
4. Bob verifies the signature and recovers the message $v_A(s_A(w)) = w$.

DIGITAL SIGNATURES with ENCRYPTION and RESENDING

1. Alice signs the message: $s_A(w)$.
2. Alice encrypts the signed message: $e_B(s_A(w))$.
3. Bob decrypts the signed message: $d_B(e_B(s_A(w))) = s_A(w)$.
4. Bob verifies the signature and recovers the message $v_A(s_A(w)) = w$.

Resending the message as a receipt

5. Bob signs and encrypts the message and sends to Alice $e_A(s_B(w))$.

DIGITAL SIGNATURES with ENCRYPTION and RESENDING

1. Alice signs the message: $s_A(w)$.
2. Alice encrypts the signed message: $e_B(s_A(w))$.
3. Bob decrypts the signed message: $d_B(e_B(s_A(w))) = s_A(w)$.
4. Bob verifies the signature and recovers the message $v_A(s_A(w)) = w$.

Resending the message as a receipt

5. Bob signs and encrypts the message and sends to Alice $e_A(s_B(w))$.
6. Alice decrypts the message and verifies the signature.

DIGITAL SIGNATURES with ENCRYPTION and RESENDING

1. Alice signs the message: $s_A(w)$.
2. Alice encrypts the signed message: $e_B(s_A(w))$.
3. Bob decrypts the signed message: $d_B(e_B(s_A(w))) = s_A(w)$.
4. Bob verifies the signature and recovers the message $v_A(s_A(w)) = w$.

Resending the message as a receipt

5. Bob signs and encrypts the message and sends to Alice $e_A(s_B(w))$.
6. Alice decrypts the message and verifies the signature.

Assume now: $v_x = e_x$, $s_x = d_x$ for all users x.

A SURPRISING ATTACK to PREVIOUS SCHEME

1 Mallot intercepts $e_B(s_A(w))$.

A SURPRISING ATTACK to PREVIOUS SCHEME

1. Mallot intercepts $e_B(s_A(w))$.
2. Later Mallot sends $e_B(s_A(w))$ to Bob pretending it is from him (from Mallot).

A SURPRISING ATTACK to PREVIOUS SCHEME

1. Mallot intercepts $e_B(s_A(w))$.
2. Later Mallot sends $e_B(s_A(w))$ to Bob pretending it is from him (from Mallot).
3. Bob decrypts and "verifies" the message by computing

$$e_M(d_B(e_B(d_A(w)))) = e_M(d_A(w)) \text{ – a garbage.}$$

A SURPRISING ATTACK to PREVIOUS SCHEME

1. Mallot intercepts $e_B(s_A(w))$.
2. Later Mallot sends $e_B(s_A(w))$ to Bob pretending it is from him (from Mallot).
3. Bob decrypts and "verifies" the message by computing

$$e_M(d_B(e_B(d_A(w)))) = e_M(d_A(w)) \text{ – a garbage.}$$

4. Bob goes on with the protocol and returns to Mallot the receipt:

$$e_M(d_B(e_M(d_A(w))))$$

A SURPRISING ATTACK to PREVIOUS SCHEME

1. Mallot intercepts $e_B(s_A(w))$.
2. Later Mallot sends $e_B(s_A(w))$ to Bob pretending it is from him (from Mallot).
3. Bob decrypts and "verifies" the message by computing

   $$e_M(d_B(e_B(d_A(w)))) = e_M(d_A(w)) \text{ -- a garbage.}$$

4. Bob goes on with the protocol and returns to Mallot the receipt:

   $$e_M(d_B(e_M(d_A(w))))$$

5. Mallot can then get w.

   Indeed, Mallot can compute

   $e_A(d_M(e_B(d_M(e_M(d_B(e_M(d_A(w)))))))) = $ w.

A MAN-IN-THE-MIDDLE ATTACK

Consider the following protocol:

1. Alice sends Bob the pair $(e_B(e_B(w)||A), B)$ to B.
2. Bob uses d_B to get A and w, and acknowledges by sending the pair $(e_A(e_A(w)||B), A)$ to Alice.

(Here the function e and d are assumed to operate on strings and identificators $A, B, \ldots$ are strings.

A MAN-IN-THE-MIDDLE ATTACK

Consider the following protocol:

1. Alice sends Bob the pair $(e_B(e_B(w)||A), B)$ to B.
2. Bob uses d_B to get A and w, and acknowledges by sending the pair $(e_A(e_A(w)||B), A)$ to Alice.

(Here the function e and d are assumed to operate on strings and identificators $A, B, \ldots$ are strings.

What can an active eavesdropper C do?

- C can learn $(e_A(e_A(w)||B), A)$ and therefore $e_A(w'), w' = e_A(w)||B$.
- C can now send to Alice the pair $(e_A(e_A||w')||C), A)$.
- Alice, thinking that this is the step 1 of the protocol, acknowledges by sending the pair $(e_C(e_C(w')||A), C)$ to C.
- C is now able to learn w' and therefore also $e_A(w)$.
- C now sends to Alice the pair $(e_A(e_A(w)||C), A)$.
- Alice acknowledges by sending the pair $(e_C(e_C(w)||A), C)$.
- C is now able to learn w.

PROBABILISTIC SIGNATURES SCHEMES - PSS

Let us have integers k, l, n such that $k + l < n$, a permutation

$$f : D \rightarrow D, D \subset \{0,1\}^n,$$

a pseudorandom bit generator

$$G : \{0,1\}^l \rightarrow \{0,1\}^k \times \{0,1\}^{n-(l+k)}, w \rightarrow (G_1(w), G_2(w))$$

and a hash function

$$h : \{0,1\}^* \rightarrow \{0,1\}^l.$$

The following PSS scheme is applicable to messages of arbitrary length.

PROBABILISTIC SIGNATURES SCHEMES - PSS

Let us have integers k, l, n such that $k + l < n$, a permutation

$$f : D \to D, D \subset \{0,1\}^n,$$

a pseudorandom bit generator

$$G : \{0,1\}^l \to \{0,1\}^k \times \{0,1\}^{n-(l+k)}, w \to (G_1(w), G_2(w))$$

and a hash function

$$h : \{0,1\}^* \to \{0,1\}^l.$$

The following PSS scheme is applicable to messages of arbitrary length.

Signing: of a message $w \in \{0,1\}^*$.

1. Choose random $r \in \{0,1\}^k$ and compute $m = h(w\|r)$.
2. Compute $G(m) = (G_1(m), G_2(m))$ and $y = m\|(G_1(m) \oplus r)\|G_2(m)$.
3. Signature of w is $\sigma = f^{-1}(y)$.

PROBABILISTIC SIGNATURES SCHEMES - PSS

Let us have integers k, l, n such that $k + l < n$, a permutation

$$f : D \to D, D \subset \{0,1\}^n,$$

a pseudorandom bit generator

$$G : \{0,1\}^l \to \{0,1\}^k \times \{0,1\}^{n-(l+k)}, w \to (G_1(w), G_2(w))$$

and a hash function

$$h : \{0,1\}^* \to \{0,1\}^l.$$

The following PSS scheme is applicable to messages of arbitrary length.

Signing: of a message $w \in \{0,1\}^*$.

1. Choose random $r \in \{0,1\}^k$ and compute $m = h(w\|r)$.
2. Compute $G(m) = (G_1(m), G_2(m))$ and $y = m\|(G_1(m) \oplus r)\|G_2(m)$.
3. Signature of w is $\sigma = f^{-1}(y)$.

Verification of a signed message (w, σ).

- Compute $f(\sigma)$ and decompose $f(\sigma) = m\|t\|u$, where $|m| = l$, $|t| = k$ and $|u| = n - (k + l)$.
- Compute $r = t \oplus G_1(m)$.
- Accept signature σ if $h(w\|r) = m$ and $G_2(m) = u$; otherwise reject it.

Diffie-Hellman PUBLIC ESTABLISHMENT of SECRET KEYS - rpetition

Main problem of the secret-key cryptography: a need to make a secure distribution (establishment) of secret keys ahead of transmissions.

Diffie+Hellman solved this problem in 1976 by designing a protocol for secure key establishment (distribution) over public channels.

Diffie-Hellman PUBLIC ESTABLISHMENT of SECRET KEYS - rpetition

Main problem of the secret-key cryptography: a need to make a secure distribution (establishment) of secret keys ahead of transmissions.

Diffie+Hellman solved this problem in 1976 by designing a protocol for secure key establishment (distribution) over public channels.

Diffie-Hellman Protocol: If two parties, Alice and Bob, want to create a common secret key, then they first agree, somehow, on a large prime p and a q<p of large order in Z_p^* and then they perform, through a public channel, the following activities.

- Alice chooses, randomly, a large $1 \leq x < p - 1$ and computes

$$X = q^x \bmod p.$$

Main problem of the secret-key cryptography: a need to make a secure distribution (establishment) of secret keys ahead of transmissions.

Diffie+Hellman solved this problem in 1976 by designing a protocol for secure key establishment (distribution) over public channels.

Diffie-Hellman Protocol: If two parties, Alice and Bob, want to create a common secret key, then they first agree, somehow, on a large prime p and a q<p of large order in Z_p^* and then they perform, through a public channel, the following activities.

- Alice chooses, randomly, a large $1 \leq x < p-1$ and computes
$$X = q^x \bmod p.$$
- Bob also chooses, again randomly, a large $1 \leq y < p-1$ and computes
$$Y = q^y \bmod p.$$

Diffie-Hellman PUBLIC ESTABLISHMENT of SECRET KEYS - rpetition

Main problem of the secret-key cryptography: a need to make a secure distribution (establishment) of secret keys ahead of transmissions.

Diffie+Hellman solved this problem in 1976 by designing a protocol for secure key establishment (distribution) over public channels.

Diffie-Hellman Protocol: If two parties, Alice and Bob, want to create a common secret key, then they first agree, somehow, on a large prime p and a q<p of large order in Z_p^* and then they perform, through a public channel, the following activities.

- Alice chooses, randomly, a large $1 \leq x < p-1$ and computes

$$X = q^x \bmod p.$$

- Bob also chooses, again randomly, a large $1 \leq y < p-1$ and computes

$$Y = q^y \bmod p.$$

- Alice and Bob exchange **X** and **Y**, through a public channel, but keep **x, y** secret.

Diffie-Hellman PUBLIC ESTABLISHMENT of SECRET KEYS - rpetition

Main problem of the secret-key cryptography: a need to make a secure distribution (establishment) of secret keys ahead of transmissions.

Diffie+Hellman solved this problem in 1976 by designing a protocol for secure key establishment (distribution) over public channels.

Diffie-Hellman Protocol: If two parties, Alice and Bob, want to create a common secret key, then they first agree, somehow, on a large prime p and a q<p of large order in Z_p^* and then they perform, through a public channel, the following activities.

- Alice chooses, randomly, a large $1 \leq x < p-1$ and computes
$$X = q^x \bmod p.$$
- Bob also chooses, again randomly, a large $1 \leq y < p-1$ and computes
$$Y = q^y \bmod p.$$
- Alice and Bob exchange **X** and **Y**, through a public channel, but keep **x, y** secret.
- Alice computes $Y^x \bmod p$ and Bob computes $X^y \bmod p$ and then each of them has the key
$$K = q^{xy} \bmod p.$$

Diffie-Hellman PUBLIC ESTABLISHMENT of SECRET KEYS - rpetition

Main problem of the secret-key cryptography: a need to make a secure distribution (establishment) of secret keys ahead of transmissions.

Diffie+Hellman solved this problem in 1976 by designing a protocol for secure key establishment (distribution) over public channels.

Diffie-Hellman Protocol: If two parties, Alice and Bob, want to create a common secret key, then they first agree, somehow, on a large prime p and a q<p of large order in Z_p^* and then they perform, through a public channel, the following activities.

- Alice chooses, randomly, a large $1 \leq x < p-1$ and computes
$$X = q^x \bmod p.$$
- Bob also chooses, again randomly, a large $1 \leq y < p-1$ and computes
$$Y = q^y \bmod p.$$
- Alice and Bob exchange **X** and **Y**, through a public channel, but keep **x, y** secret.
- Alice computes $Y^x \bmod p$ and Bob computes $X^y \bmod p$ and then each of them has the key
$$K = q^{xy} \bmod p.$$

Main problem of the secret-key cryptography: a need to make a secure distribution (establishment) of secret keys ahead of transmissions.

Diffie+Hellman solved this problem in 1976 by designing a protocol for secure key establishment (distribution) over public channels.

Diffie-Hellman Protocol: If two parties, Alice and Bob, want to create a common secret key, then they first agree, somehow, on a large prime p and a q<p of large order in Z_p^* and then they perform, through a public channel, the following activities.

- Alice chooses, randomly, a large $1 \leq x < p-1$ and computes
$$X = q^x \bmod p.$$
- Bob also chooses, again randomly, a large $1 \leq y < p-1$ and computes
$$Y = q^y \bmod p.$$
- Alice and Bob exchange **X** and **Y**, through a public channel, but keep **x, y** secret.
- Alice computes $Y^x \bmod p$ and Bob computes $X^y \bmod p$ and then each of them has the key
$$K = q^{xy} \bmod p.$$

An eavesdropper seems to need, in order to determine x from **X, q, p** and y from **Y, q, p**, a capability to compute discrete logarithms, or to compute q^{xy} from q^x and q^y, what is believed to be infeasible.

AUTHENTICATED Diffie-Hellman KEY EXCHANGE

Let each user U has a signature algorithm s_U and a verification algorithm v_U.
The following protocol allows Alice and Bob to establish a key K to use with an encryption function e_K and to avoid the man-in-the-middle attack.

1. Alice and Bob choose large prime p and a generator $q \in Z_p^*$.

AUTHENTICATED Diffie-Hellman KEY EXCHANGE

Let each user U has a signature algorithm s_U and a verification algorithm v_U.
The following protocol allows Alice and Bob to establish a key K to use with an encryption function e_K and to avoid the man-in-the-middle attack.

1. Alice and Bob choose large prime p and a generator $q \in Z_p^*$.
2. Alice chooses a random x and Bob chooses a random y.
3. Alice computes $q^x \mod p$, and Bob computes $q^y \mod p$.
4. Alice sends q^x to Bob.
5. Bob computes $K = q^{xy} \mod p$.
6. Bob sends q^y and $e_K(s_B(q^y, q^x))$ to Alice.
7. Alice computes $K = q^{xy} \mod p$.
8. Alice decrypts $e_K(s_B(q^y, q^x))$ to obtain $s_B(q^y, q^x)$.
9. Alice verifies, using an authority, that v_B is Bob's verification algorithm.
10. Alice uses v_B to verify Bob's signature.
11. Alice sends $e_K(s_A(q^x, q^y))$ to Bob.
12. Bob decrypts, verifies v_A, and verifies Alice's signature.

An enhanced version of the above protocol is known as Station-to-Station protocol.

THRESHOLD DIGITAL SIGNATURES

The idea of a (t+1, n) threshold signature scheme is to distribute the power of the signing operation to (t+1) parties out of n.

A (t+1) threshold signature scheme should satisfy two conditions.

Unforgeability means that even if an adversary corrupts t parties, he still cannot generate a valid signature.

Robustness means that corrupted parties cannot prevent uncorrupted parties to generate signatures.

Shoup (2000) presented an efficient, non-interactive, robust and unforgeable threshold RSA signature schemes.

- In 1976 Diffie and Hellman were first to describe the idea of a digital signature scheme. However, they only conjectured that such schemes may exist.
- In 1977 RSA was invented that could be used to produce a primitive (not secure enough) digital signatures.
- The first widely marketed software package to offer digital signature was Lotus Notes 1.0, based on RSA and released in 1989
- ElGamal diital signatures were invented in 1984.
- In 1988 Goldwasser, Micali and Rivest were first to rigorously define (perfect0 security of digital signature schemes.

APPENDIX to CHAPTER 7

SPECIAL TYPES of DIGITAL SIGNATURES

- Append-Only Signatures (AOS) have the property that any party given an AOS signature $sig[M_1]$ on message M_1 can compute $sig[M_1 \| M_2]$ for any message M_2. (Such signatures are of importance in network applications, where users need to delegate their shares of resources to other users).
- Identity-Based signatures (IBS) at which the identity of the signer (i.e. her email address) plays the role of her public key. (Such schemes assume the existence of a TA holding a master public-private key pair used to assign secret keys to users based on their identity.)
- Hierarchically Identity-Based Signatures are such IBS in which users are arranged in a hierarchy and a user at any level at the hierarchy can delegate secret keys to her descendants based on their identities and her own secret keys.

GROUP SIGNATURES

- At Group Signatures (GS) a group member can compute a signature that reveals nothing about the signer's identity, except that he is a member of the group. On the other hand, the group manager can always reveal the identity of the signer.

- Hierarchical Group Signatures (HGS) are a generalization of GS that allow multiple group managers to be organized in a tree with the signers as leaves. When verifying a signature, a group manager only learns to which of its subtrees, if any, the signer belongs.

UNCONDITIONALLY SECURE DIGITAL SIGNATURES

Any of the digital signature schemes introduced so far can be forged by anyone having enough computer power.

Chaum and Roijakkers (2001) developed, for any fixed set of users, an unconditionally secure signature scheme with the following properties:

- Any participant can convince (except with exponentially small probability) any other participant that his signature is valid.
- A convinced participant can convince any other participant of the signature's validity, without interaction with the original signer.

Assume Alice uses a hash function that produces 50 bits.

Fred, who wants Alice to sign a fraudulent contract, find 30 places in a good document, where he can make change in the document (adding a coma, space, ...) such that Alice would not notice that. By choosing at each place whether to make or not a change, he can produce 2^{30} documents essentially identical with the original good document.

BIRTHDAY PARADOX ATTACK on DIGITAL SIGNATURE

Assume Alice uses a hash function that produces 50 bits.

Fred, who wants Alice to sign a fraudulent contract, find 30 places in a good document, where he can make change in the document (adding a coma, space, ...) such that Alice would not notice that. By choosing at each place whether to make or not a change, he can produce 2^{30} documents essentially identical with the original good document.

Similarly, Fred makes 2^{30} changes of the fraudulent document.

Considering birthday problem with $n = 2^{50}$, $r = 2^{30}$ we get that $r = \sqrt{\lambda n}$, with $\lambda = 2^{10}$ and therefore with probability $1 - e^{-1024} \approx 1$ there is a version of the good document that has the same hash as a version of the fraudulent document.

Finding a match, Fred can ask Alice to sign a good version and then append the signature to the fraudulent contract.

- **We say that an encryption system has been broken if one can determine a plaintext from a cryptotext (often).**
- **A digital signature system is considered as broken if one can (often) forge signatures.**
- **In both cases, a more ambitious goal is to find the private key.**

RSA BASED SIGNATURES - CHOICE of PUBLIC EXPONENT

The common choice of a public exponent e is

$$3$$

or

$$2^{16}+1$$

When the value $2^{16}+1$ is used, signature verification requires 17 multiplications, as opposed to roughly 1000 when a random $e \leq O(n)$ is used.

UNDENIABLE SIGNATURES I

Undeniable signatures are signatures that have two properties:

- A signature can be verified only in the cooperation with the signer – by means of a challenge-and-response protocol.
- The signer cannot deny a correct signature. To achieve that, steps are a part of the protocol that force the signer to cooperate – by means of a disavowal protocol – this protocol makes possible to prove the invalidity of a signature and to show that it is a forgery. (If the signer refuses to take part in the disavowal protocol, then the signature is considered to be genuine.)

Undeniable signature protocol of Chaum and van Antwerpen (1989), discussed next, is again based on infeasibility of the computation of the discrete logarithm.

UNDENIABLE SIGNATURES II

Undeniable signatures consist of:

- Signing algorithm
- Verification protocol, that is a challenge-and-response protocol.

  In this case it is required that a signature cannot be verified without a cooperation of the signer (Bob).

  This protects Bob against the possibility that documents signed by him are duplicated and distributed without his approval.
- Disavowal protocol, by which Bob can prove that a signature is a forgery.

  This is to prevent Bob from disavowing a signature he made at an earlier time.

UNDENIABLE SIGNATURES II

Undeniable signatures consist of:

- Signing algorithm
- Verification protocol, that is a challenge-and-response protocol.

  In this case it is required that a signature cannot be verified without a cooperation of the signer (Bob).

  This protects Bob against the possibility that documents signed by him are duplicated and distributed without his approval.
- Disavowal protocol, by which Bob can prove that a signature is a forgery.

  This is to prevent Bob from disavowing a signature he made at an earlier time.

Chaum-van Antwerpen undeniable signature schemes (CAUSS)

- p, r are primes $p = 2r + 1$
- $q \in Z_p^*$ is of order r;
- $1 \leq x \leq r - 1$, $y = q^x \mod p$;
- G is a multiplicative subgroup of Z_p^* of order q (G consists of quadratic residues modulo p).

Key space: $K = \{p, q, x, y\}$; p, q, y are public, $x \in G$ is secret.

Signature: $s = sig_K(w) = w^x \mod p$.

FOOLING and DISALLOWED PROTOCOL I

Since it holds:

Theorem If $s \neq w^x \mod p$, then Alice will accept s as a valid signature for w with probability 1/r.

Bob cannot fool Alice except with very small probability and security is unconditional (that is, it does not depend on any computational assumption).

FOOLING and DISALLOWED PROTOCOL I

Since it holds:

Theorem If $s \neq w^x \mod p$, then Alice will accept s as a valid signature for w with probability 1/r.

Bob cannot fool Alice except with very small probability and security is unconditional (that is, it does not depend on any computational assumption).

Disallowed protocol

Basic idea: After receiving a signature s Alice initiates two independent and unsuccessful runs of the verification protocol. Finally, she performs a “consistency check” to determine whether Bob has formed his responses according to the protocol.

- Alice chooses $e_1, e_2 \in Z_r^*$.
- Alice computes $c = s^{e1}y^{e2} \mod p$ and sends it to Bob.
- Bob computes $d = c^{x^{(-1)} \mod r} \mod p$ and sends it to Alice.
- Alice verifies that $d \neq w^{e1}q^{e2} \pmod p$.
- Alice chooses $f_1, f_2 \in Z_r^*$.
- Alice computes $C = s^{f1}y^{f2} \mod p$ and sends it to Bob.
- Bob computes $D = C^{x^{(-1)} \mod r} \mod p$ and sends it to Alice.

FOOLING and DISALLOWED PROTOCOL II

- Alice verifies that $D \neq w^{f1}q^{f2} \pmod{p}$.
- Alice concludes that s is a forgery iff

$$(dq^{-e2})^{f1} \equiv (Dq^{-f2})^{e1} \pmod{p}.$$

FOOLING and DISALLOWED PROTOCOL II

- Alice verifies that $D \neq w^{f1}q^{f2} \pmod{p}$.
- Alice concludes that s is a forgery iff

$$(dq^{-e2})^{f1} \equiv (Dq^{-f2})^{e1} \pmod{p}.$$

CONCLUSIONS

It can be shown:

Bob can convince Alice that an invalid signature is a forgery. In order to do that it is sufficient to show that if $s \neq w^x$, then

$$(dq^{-e2})^{f1} \equiv (Dq^{-f2})^{e1} \pmod{p}$$

what can be done using congruency relation from the design of the signature system and from the disallowed protocol.

Bob cannot make Alice believe that a valid signature is a forgery, except with a very small probability.

Part VIII

Elliptic curves cryptography and factorization

ELLIPTIC CURVE CRYPTOGRAPHY and FACTORIZATION

Cryptography based on manipulation of points of so called **elliptic curves** is currently getting momentum and has a tendency to replace public key cryptography based on the infeasibility of factorization of integers, or on infeasibility of the computation of discrete logarithms.

ELLIPTIC CURVE CRYPTOGRAPHY and FACTORIZATION

Cryptography based on manipulation of points of so called **elliptic curves** is currently getting momentum and has a tendency to replace public key cryptography based on the infeasibility of factorization of integers, or on infeasibility of the computation of discrete logarithms.

For example, the US-government has recommended to its governmental institutions to use mainly **elliptic curve cryptography - ECC.**

ELLIPTIC CURVE CRYPTOGRAPHY and FACTORIZATION

Cryptography based on manipulation of points of so called **elliptic curves** is currently getting momentum and has a tendency to replace public key cryptography based on the infeasibility of factorization of integers, or on infeasibility of the computation of discrete logarithms.

For example, the US-government has recommended to its governmental institutions to use mainly **elliptic curve cryptography - ECC.**

The main advantage of elliptic curves cryptography is that to achieve a certain level of security shorter keys are sufficient than in case of "usual cryptography". Using shorter keys can result in a considerable savings in hardware implementations.

ELLIPTIC CURVE CRYPTOGRAPHY and FACTORIZATION

Cryptography based on manipulation of points of so called **elliptic curves** is currently getting momentum and has a tendency to replace public key cryptography based on the infeasibility of factorization of integers, or on infeasibility of the computation of discrete logarithms.

For example, the US-government has recommended to its governmental institutions to use mainly **elliptic curve cryptography - ECC.**

The main advantage of elliptic curves cryptography is that to achieve a certain level of security shorter keys are sufficient than in case of "usual cryptography". Using shorter keys can result in a considerable savings in hardware implementations.

The second advantage of the elliptic curves cryptography is that quite a few of attacks developed for cryptography based on factorization and discrete logarithm do not work for the elliptic curves cryptography.

ELLIPTIC CURVE CRYPTOGRAPHY and FACTORIZATION

Cryptography based on manipulation of points of so called **elliptic curves** is currently getting momentum and has a tendency to replace public key cryptography based on the infeasibility of factorization of integers, or on infeasibility of the computation of discrete logarithms.

For example, the US-government has recommended to its governmental institutions to use mainly **elliptic curve cryptography - ECC.**

The main advantage of elliptic curves cryptography is that to achieve a certain level of security shorter keys are sufficient than in case of "usual cryptography". Using shorter keys can result in a considerable savings in hardware implementations.

The second advantage of the elliptic curves cryptography is that quite a few of attacks developed for cryptography based on factorization and discrete logarithm do not work for the elliptic curves cryptography.

It is amazing how practical is the elliptic curve cryptography that is based on very strangely looking theoretical concepts.

ELLIPTIC CURVES

An elliptic curve E is the graph of the relation defined by the equation

$$E : y^2 = x^3 + ax + b$$

(where a, b are either rational numbers or integers (and computation is done modulo some integer n)) extended by a "point at infinity", denoted usually as ∞ (or 0) that can be regarded as being, at the same time, at the very top and very bottom of the y-axis.

An elliptic curve E is the graph of the relation defined by the equation

$$E : y^2 = x^3 + ax + b$$

(where a, b are either rational numbers or integers (and computation is done modulo some integer n)) extended by a "point at infinity", denoted usually as ∞ (or 0) that can be regarded as being, at the same time, at the very top and very bottom of the y-axis.

We will consider mainly only those elliptic curves that have no multiple roots - which is equivalent to the condition $4a^3 + 27b^2 \neq 0$.

ELLIPTIC CURVES

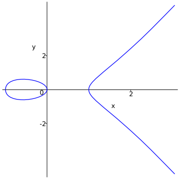
An elliptic curve E is the graph of the relation defined by the equation

$$E : y^2 = x^3 + ax + b$$

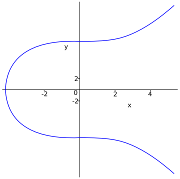
(where a, b are either rational numbers or integers (and computation is done modulo some integer n)) extended by a "point at infinity", denoted usually as ∞ (or 0) that can be regarded as being, at the same time, at the very top and very bottom of the y-axis.

We will consider mainly only those elliptic curves that have no multiple roots - which is equivalent to the condition $4a^3 + 27b^2 \neq 0$.

In case coefficients and x, y can be any rational numbers, a graph of an elliptic curve has one of the forms shown in the following figure. The graph depends on whether the polynomial $x^3 + ax + b$ has three or only one real root.



$y^2 = x(x+1)(x-1)$



$y^2 = x^3 + 73$

Elliptic curves are not ellipses and therefore it seems strange that they have such a name.

HISTORICAL REMARKS on ELLIPTIC CURVES

Elliptic curves are not ellipses and therefore it seems strange that they have such a name. Elliptic curves actually received their names from their relation to so called elliptic integrals

$$\int_{x1}^{x2} \frac{dx}{\sqrt{x^3 + ax + b}} \qquad \int_{x1}^{x2} \frac{xdx}{\sqrt{x^3 + ax + b}}$$

that arise in the computation of the arc-length of ellipses.

HISTORICAL REMARKS on ELLIPTIC CURVES

Elliptic curves are not ellipses and therefore it seems strange that they have such a name. Elliptic curves actually received their names from their relation to so called elliptic integrals

$$\int_{x1}^{x2} \frac{dx}{\sqrt{x^3 + ax + b}} \qquad \int_{x1}^{x2} \frac{xdx}{\sqrt{x^3 + ax + b}}$$

that arise in the computation of the arc-length of ellipses.

It may also seem puzzling why not to consider curves given by more general equations

$$y^2 + cxy + dy = x^3 + ex^2 + ax + b$$

HISTORICAL REMARKS on ELLIPTIC CURVES

Elliptic curves are not ellipses and therefore it seems strange that they have such a name. Elliptic curves actually received their names from their relation to so called elliptic integrals

$$\int_{x1}^{x2} \frac{dx}{\sqrt{x^3 + ax + b}} \qquad \int_{x1}^{x2} \frac{xdx}{\sqrt{x^3 + ax + b}}$$

that arise in the computation of the arc-length of ellipses.

It may also seem puzzling why not to consider curves given by more general equations

$$y^2 + cxy + dy = x^3 + ex^2 + ax + b$$

The reason is that if we are working with rational coefficients or mod p, where $p > 3$ is a prime, then such a general equation can be transformed to our special case of equation. In other cases, it may be necessary to consider the most general form of equation.

Geometry

On any elliptic curve we can define addition of points in such a way that points of the corresponding curve with such an operation of addition form an Abelian group. in which ∞ point is the identity element

Geometry

On any elliptic curve we can define addition of points in such a way that points of the corresponding curve with such an operation of addition form an Abelian group. in which ∞ point is the identity element

If the line through two different points P_1 and P_2 of an elliptic curve E intersects E in a point $Q = (x, y)$, then we define $P_1 + P_2 = P_3 = (x, -y)$. (This also implies that for any point P on E it holds $P + \infty = P + 0P$.) ∞ therefore play a role of null element

Geometry

On any elliptic curve we can define addition of points in such a way that points of the corresponding curve with such an operation of addition form an Abelian group. in which ∞ point is the identity element

If the line through two different points P_1 and P_2 of an elliptic curve E intersects E in a point $Q = (x, y)$, then we define $P_1 + P_2 = P_3 = (x, -y)$. (This also implies that for any point P on E it holds $P + \infty = P + 0P$.) ∞ therefore play a role of null element
If the line through two different points P_1 and P_2 is parallel with y-axis, then we define $P_1 + P_2 = \infty$.

Geometry

On any elliptic curve we can define addition of points in such a way that points of the corresponding curve with such an operation of addition form an Abelian group. in which ∞ point is the identity element

If the line through two different points P_1 and P_2 of an elliptic curve E intersects E in a point $Q = (x, y)$, then we define $P_1 + P_2 = P_3 = (x, -y)$. (This also implies that for any point P on E it holds $P + \infty = P + 0P$.) ∞ therefore play a role of null element
If the line through two different points P_1 and P_2 is parallel with y-axis, then we define $P_1 + P_2 = \infty$.
In case $P_1 = P_2$, and the tangent to E in P_1 intersects E in a point $Q = (x, y)$, then we define $P_1 + P_1 = (x, -y)$.

Geometry

On any elliptic curve we can define addition of points in such a way that points of the corresponding curve with such an operation of addition form an Abelian group. in which ∞ point is the identity element

If the line through two different points P_1 and P_2 of an elliptic curve E intersects E in a point $Q = (x, y)$, then we define $P_1 + P_2 = P_3 = (x, -y)$. (This also implies that for any point P on E it holds $P + \infty = P + 0P$.) ∞ therefore play a role of null element
If the line through two different points P_1 and P_2 is parallel with y-axis, then we define $P_1 + P_2 = \infty$.
In case $P_1 = P_2$, and the tangent to E in P_1 intersects E in a point $Q = (x, y)$, then we define $P_1 + P_1 = (x, -y)$.
It should now be obvious how to define subtraction of two points of an elliptic curve.

ADDITION of POINTS on ELLIPTIC CURVES - GEOMETRY (1)

Geometry

On any elliptic curve we can define addition of points in such a way that points of the corresponding curve with such an operation of addition form an Abelian group. in which ∞ point is the identity element

If the line through two different points P_1 and P_2 of an elliptic curve E intersects E in a point $Q = (x, y)$, then we define $P_1 + P_2 = P_3 = (x, -y)$. (This also implies that for any point P on E it holds $P + \infty = P + 0P$.) ∞ therefore play a role of null element

If the line through two different points P_1 and P_2 is parallel with y-axis, then we define $P_1 + P_2 = \infty$.

In case $P_1 = P_2$, and the tangent to E in P_1 intersects E in a point $Q = (x, y)$, then we define $P_1 + P_1 = (x, -y)$.

It should now be obvious how to define subtraction of two points of an elliptic curve.

It is now easy to verify that the above addition of points forms Abelian group with ∞ as the identity (null) element.

ELLIPTIC CURVES - GENERALITY

A general elliptic curve over Z_{p^m} where p is a prime is the set of points (x, y) satisfying so-called Weierstrass equation

$$y^2 + uxy + vy = x^3 + ax^2 + bx + c$$

for some constants u, v, a, b, c together with a single element $\mathbf{0}$, called the point of infinity.

ELLIPTIC CURVES - GENERALITY

A general elliptic curve over Z_{p^m} where p is a prime is the set of points (x, y) satisfying so-called Weierstrass equation

$$y^2 + uxy + vy = x^3 + ax^2 + bx + c$$

for some constants u, v, a, b, c together with a single element $\mathbf{0}$, called the point of infinity.

If $p \neq 2$ Weierstrass equation can be simplified by transformation

$$y \rightarrow \frac{y - (ux + v)}{2}$$

to get the equation

$$y^2 = x^3 + dx^2 + ex + f$$

for some constants d, e, f and if $p \neq 3$ by transformation

$$x \rightarrow x - \frac{d}{3}$$

to get equation

$$y^2 = x^3 + fx + g$$

Formulas

Addition of points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ of an elliptic curve $E : y^2 = x^3 + ax + b$ can be easily computed using the following formulas:

$$P_1 + P_2 = P_3 = (x_3, y_3)$$

where

$$x_3 = \lambda^2 - x_1 - x_2$$
$$y_3 = \lambda(x_1 - x_3) - y_1$$

and

ADDITION of POINTS on ELLIPTIC CURVES (2)

Formulas

Addition of points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ of an elliptic curve $E : y^2 = x^3 + ax + b$ can be easily computed using the following formulas:

$$P_1 + P_2 = P_3 = (x_3, y_3)$$

where

$$x_3 = \lambda^2 - x_1 - x_2$$
$$y_3 = \lambda(x_1 - x_3) - y_1$$

and

$$\lambda = \begin{cases} \frac{(y_2 - y_1)}{(x_2 - x_1)} & \text{if } P_1 \neq P_2, \\ \frac{(3x_1^2 + a)}{(2y_1)} & \text{if } P_1 = P_2. \end{cases}$$

All that holds for the case that $\lambda \neq \infty$; otherwise $P_3 = \infty$.

Formulas

Addition of points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ of an elliptic curve $E : y^2 = x^3 + ax + b$ can be easily computed using the following formulas:

$$P_1 + P_2 = P_3 = (x_3, y_3)$$

where

$$x_3 = \lambda^2 - x_1 - x_2$$
$$y_3 = \lambda(x_1 - x_3) - y_1$$

and

$$\lambda = \begin{cases} \frac{(y_2 - y_1)}{(x_2 - x_1)} & \text{if } P_1 \neq P_2, \\ \frac{(3x_1^2 + a)}{(2y_1)} & \text{if } P_1 = P_2. \end{cases}$$

All that holds for the case that $\lambda \neq \infty$; otherwise $P_3 = \infty$.

Example For curve $y^2 = x^3 + 73$ and $P_1 = (2, 9)$, $P_2 = (3, 10)$ we have $\lambda = 1$, $P_1 + P_2 = P_3 = (-4, -3)$ and $P_3 + P_3 = (72, 611)$.

ELLIPTIC CURVES mod n

The points on an elliptic curve

$$E : y^2 = x^3 + ax + b \pmod{n}$$

are such pairs (x,y) mod n that satisfy the above equation, along with the point ∞ at infinity.

ELLIPTIC CURVES mod n

The points on an elliptic curve

$$E : y^2 = x^3 + ax + b \pmod n$$

are such pairs (x,y) mod n that satisfy the above equation, along with the point ∞ at infinity.

Example Elliptic curve $E : y^2 = x^3 + 2x + 3 \pmod 5$ has points

$$(1,1),(1,4),(2,0),(3,1),(3,4),(4,0),\infty.$$

ELLIPTIC CURVES mod n

The points on an elliptic curve

$$E: y^2 = x^3 + ax + b \pmod n$$

are such pairs (x,y) mod n that satisfy the above equation, along with the point ∞ at infinity.

Example Elliptic curve $E: y^2 = x^3 + 2x + 3 \pmod 5$ has points

$$(1,1),(1,4),(2,0),(3,1),(3,4),(4,0),\infty.$$

Example For elliptic curve $E: y^2 = x^3 + x + 6$ (mod 11) and its point $P = (2,7)$ it holds $2P = (5,2)$; $3P = (8,3)$. Number of points on an elliptic curve (mod p) can be easily estimated.

ELLIPTIC CURVES mod n

The points on an elliptic curve

$$E : y^2 = x^3 + ax + b \pmod n$$

are such pairs (x,y) mod n that satisfy the above equation, along with the point ∞ at infinity.

Example Elliptic curve $E : y^2 = x^3 + 2x + 3 \pmod 5$ has points

$$(1,1),(1,4),(2,0),(3,1),(3,4),(4,0),\infty.$$

Example For elliptic curve $E : y^2 = x^3 + x + 6 \; (\mod 11)$ and its point $P = (2,7)$ it holds $2P = (5,2)$; $3P = (8,3)$. Number of points on an elliptic curve (mod p) can be easily estimated.

Hasse's theorem If an elliptic curve $E(\bmod p)$ has $|E|$ points then $|p - 1| < 2\sqrt{p}$

ELLIPTIC CURVES mod n

The points on an elliptic curve

$$E : y^2 = x^3 + ax + b \pmod n$$

are such pairs (x,y) mod n that satisfy the above equation, along with the point ∞ at infinity.

Example Elliptic curve $E : y^2 = x^3 + 2x + 3 \pmod 5$ has points

$$(1,1),(1,4),(2,0),(3,1),(3,4),(4,0),\infty.$$

Example For elliptic curve $E : y^2 = x^3 + x + 6 \;(\mod 11)$ and its point $P = (2,7)$ it holds $2P = (5,2)$; $3P = (8,3)$. Number of points on an elliptic curve (mod p) can be easily estimated.

Hasse's theorem If an elliptic curve $E(\bmod p)$ has $|E|$ points then $|p-1| < 2\sqrt{p}$

The addition of points on an elliptic curve mod n is done by the same formulas as given previously, except that instead of rational numbers c/d we deal with cd^{-1}

ELLIPTIC CURVES mod n

The points on an elliptic curve

$$E : y^2 = x^3 + ax + b \pmod n$$

are such pairs (x,y) mod n that satisfy the above equation, along with the point ∞ at infinity.

Example Elliptic curve $E : y^2 = x^3 + 2x + 3 \pmod 5$ has points

$$(1, 1), (1, 4), (2, 0), (3, 1), (3, 4), (4, 0), \infty.$$

Example For elliptic curve $E : y^2 = x^3 + x + 6 \;(\mod 11)$ and its point $P = (2, 7)$ it holds $2P = (5, 2)$; $3P = (8, 3)$. Number of points on an elliptic curve (mod p) can be easily estimated.

Hasse's theorem If an elliptic curve $E(\mod p)$ has $|E|$ points then $|p - 1| < 2\sqrt{p}$

The addition of points on an elliptic curve mod n is done by the same formulas as given previously, except that instead of rational numbers c/d we deal with cd^{-1}

Example For the curve $E : y^2 = x^3 + 2x + 3$ it holds
$(1, 4) + (3, 1) = (2, 0); (1, 4) + (2, 0) = (?, ?)$.

ELLIPTIC CURVES DISCRETE LOGARITHM

Let E be an elliptic curve and A, B be its points such that $B = kA = (A + A + \ldots A + A)$ – k times – for some k. The task to find such a k is called the discrete logarithm problem for elliptic curves.

ELLIPTIC CURVES DISCRETE LOGARITHM

Let E be an elliptic curve and A, B be its points such that $B = kA = (A + A + \ldots A + A)$ – k times – for some k. The task to find such a k is called the discrete logarithm problem for elliptic curves.

No efficient algorithm to compute discrete logarithm problem for elliptic curves is known and also no good general attacks. Elliptic curves based cryptography is based on these facts.

ELLIPTIC CURVES DISCRETE LOGARITHM

Let E be an elliptic curve and A, B be its points such that $B = kA = (A + A + \ldots A + A)$ – k times – for some k. The task to find such a k is called the discrete logarithm problem for elliptic curves.

No efficient algorithm to compute discrete logarithm problem for elliptic curves is known and also no good general attacks. Elliptic curves based cryptography is based on these facts.

There is the following general procedure for changing a discrete logarithm based cryptographic protocols to a cryptographic protocols based on elliptic curves:

- Assign to the message (plaintext) a point on an elliptic curve.
- Change, in the cryptographic protocol, modular multiplication to addition of points on an elliptic curve.
- Change, in the cryptographic protocol, exponentiation to multiplication of a point on the elliptic curve by an integer.
- To the point of an elliptic curve that results from such a protocol one assigns a message (cryptotext).

MAPPING MESSAGES into POINTS of ELLIPTIC CURVES (I)

Problem and basic idea

The problem of assigning messages to points on elliptic curves is difficult because there are no polynomial-time algorithms to write down points of an arbitrary elliptic curve.

Problem and basic idea

The problem of assigning messages to points on elliptic curves is difficult because there are no polynomial-time algorithms to write down points of an arbitrary elliptic curve.

Fortunately, there is a fast randomized algorithm, to assign points of any elliptic curve to messages, that can fail with probability that can be made arbitrarily small.

MAPPING MESSAGES into POINTS of ELLIPTIC CURVES (I)

Problem and basic idea

The problem of assigning messages to points on elliptic curves is difficult because there are no polynomial-time algorithms to write down points of an arbitrary elliptic curve.

Fortunately, there is a fast randomized algorithm, to assign points of any elliptic curve to messages, that can fail with probability that can be made arbitrarily small.

Basic idea: Given an elliptic curve $E(\bmod p)$, the problem is that not to every x there is an y such that (x, y) is a point of E.

Given a message (number) m we therefore adjoin to m few bits at the end of m and adjust them until we get a number x such that $x^3 + ax + b$ is a square $\bmod p$.

Technicalities

Let K be a large integer such that a failure rate of $\frac{1}{2^K}$ is acceptable when trying to encode a message by a point.

For $j \in \{0, \ldots, K-1\}$ verify whether for $x = mK + j$, $x^3 + ax + b \pmod p$ is a square (mod p) of an integer y.

If such an j is found, encoding is done; if not the algorithm fails (with probability $\frac{1}{2^K}$ because $x^3 + ax + b$ is a square approximately half of the time).

In order to recover the message m from the point (x, y), we compute:

$$\left\lfloor \frac{x}{K} \right\rfloor$$

ELLIPTIC CURVES KEY EXCHANGE

Elliptic curve version of the Diffie-Hellman key generation protocol goes as follows:

Let Alice and Bob agree on a prime p, on an elliptic curve E (mod p) and on a point P on E.

- Alice chooses an integer n_a, computes $n_a P$ and sends it to Bob.
- Bob chooses an integer n_b, computes $n_b P$ and sends it to Alice.
- Alice computes $n_a(n_b P)$ and Bob computes $n_b(n_a P)$. This way they have the same key.

ELLIPTIC CURVES VERSION of ElGamal CRYPTOSYSTEM

Standard version of ElGamal: Bob chooses a prime p, a generator $q < p$, an integer x, computes $y = q^x \pmod{p}$, makes public p, q, y and keeps x secret.

To send a message m Alice chooses a random r, computes:

$$a = q^r \ ; \ b = my^r$$

and sends it to Bob who decrypts by calculating $m = ba^{-x} \pmod{p}$

ELLIPTIC CURVES VERSION of ElGamal CRYPTOSYSTEM

Standard version of ElGamal: Bob chooses a prime p, a generator $q < p$, an integer x, computes $y = q^x \pmod{p}$, makes public p, q, y and keeps x secret.

To send a message m Alice chooses a random r, computes:

$$a = q^r \; ; \; b = my^r$$

and sends it to Bob who decrypts by calculating $m = ba^{-x} \pmod{p}$

Elliptic curve version of ElGamal: Bob chooses a prime p, an elliptic curve E (mod p), a point P on E, an integer x, computes Q = xP, makes E, p, and Q public and keeps x secret.

ELLIPTIC CURVES VERSION of ElGamal CRYPTOSYSTEM

Standard version of ElGamal: Bob chooses a prime p, a generator $q < p$, an integer x, computes $y = q^x \pmod{p}$, makes public p, q, y and keeps x secret.

To send a message m Alice chooses a random r, computes:

$$a = q^r \; ; \; b = my^r$$

and sends it to Bob who decrypts by calculating $m = ba^{-x} \pmod{p}$

Elliptic curve version of ElGamal: Bob chooses a prime p, an elliptic curve E (mod p), a point P on E, an integer x, computes Q = xP, makes E, p, and Q public and keeps x secret.

To send a message m ALice expresses m as a point X on E, chooses random r, computes

$$a = rP \; ; \; b = X + rQ$$

And sends the pair (a, b) to Bob who decrypts by calculating $X = b - xa$.

ELLIPTIC CURVES DIGITAL SIGNATURES

Elliptic curves version of ElGamal digital signatures has the following form for signing (a message) m, an integer, by Alice and to have the signature verified by Bob:

Alice chooses p and an elliptic curve E (mod p), a point P on E and calculates the number of points n on E (mod p) – what can be done, and we assume that $0 < m < n$.

Alice then chooses a random integer a and computes $Q = aP$. She makes public p, E, P, Q and keeps secret a.

ELLIPTIC CURVES DIGITAL SIGNATURES

Elliptic curves version of ElGamal digital signatures has the following form for signing (a message) m, an integer, by Alice and to have the signature verified by Bob:

Alice chooses p and an elliptic curve E (mod p), a point P on E and calculates the number of points n on E (mod p) – what can be done, and we assume that $0 < m < n$.

Alice then chooses a random integer a and computes Q = aP. She makes public p, E, P, Q and keeps secret a.

To sign m Alice does the following:

- Alice chooses a random integer $r, 1 \leq r < n$ such that gcd(r,n) = 1 and computes R = rP = (x,y).
- Alice computes $s = r^{-1}(m - ax) \pmod{n}$
- Alice sends the signed message (m,R,s) to Bob.

ELLIPTIC CURVES DIGITAL SIGNATURES

Elliptic curves version of ElGamal digital signatures has the following form for signing (a message) m, an integer, by Alice and to have the signature verified by Bob:

Alice chooses p and an elliptic curve E (mod p), a point P on E and calculates the number of points n on E (mod p) – what can be done, and we assume that $0 < m < n$.

Alice then chooses a random integer a and computes Q = aP. She makes public p, E, P, Q and keeps secret a.

To sign m Alice does the following:

- Alice chooses a random integer $r, 1 \leq r < n$ such that gcd(r,n) = 1 and computes R = rP = (x,y).
- Alice computes $s = r^{-1}(m - ax) \pmod{n}$
- Alice sends the signed message (m,R,s) to Bob.

Bob verifies the signature as follows:

- Bob declares the signature as valid if xQ + sR = mP

  The verification procedure works because

$$xQ + sR = xaP + r^{-1}(m - ax)(rP) = xaP + (m - ax)P = mP$$

Elliptic curves version of ElGamal digital signatures has the following form for signing (a message) m, an integer, by Alice and to have the signature verified by Bob:

Alice chooses p and an elliptic curve E (mod p), a point P on E and calculates the number of points n on E (mod p) – what can be done, and we assume that $0 < m < n$.

Alice then chooses a random integer a and computes Q = aP. She makes public p, E, P, Q and keeps secret a.

To sign m Alice does the following:

- Alice chooses a random integer $r, 1 \leq r < n$ such that gcd(r,n) = 1 and computes R = rP = (x,y).
- Alice computes $s = r^{-1}(m - ax) \pmod{n}$
- Alice sends the signed message (m,R,s) to Bob.

Bob verifies the signature as follows:

- Bob declares the signature as valid if xQ + sR = mP

  The verification procedure works because

$$xQ + sR = xaP + r^{-1}(m - ax)(rP) = xaP + (m - ax)P = mP$$

Warning Observe that actually $rr^{-1} = 1 + tn$ for some t. For the above verification procedure to work we then have to use the fact that $nP = \infty$ and therefore $P + t \cdot \infty = P$

COMMENT

Federal (USA) elliptic curve digital signature standard (ECDSA) was introduced in 20??.

DOMAIN PARAMETERS for ELLIPTIC CURVES

To use ECC all parties involved have to agree on all basic elements concerning the elliptic curve E being used:

- A prime p.
- Constants a and b in the equation $y^2 = x^3 + ax + b$.
- Generator G of the underlying cyclic subgroup such that its order is prime.
- The order n of G, that is such an n that $nG = 0$
- Co-factor $h = \frac{|E|}{n}$ that should be small $(h \leq 4)$ and, preferably $h = 1$.

To determine domain parameters (especially n and h) may be much time consuming task.

FACTORING with ELLIPTIC CURVES

Basis idea: To factorize an integer n choose an elliptic curve E, a point P on E and compute, modulo n, either iP for $i = 2, 3, 4, \ldots$ or $2^j P$ for $j = 1, 2, \ldots$. The point is that in doing that one needs to compute gcd(k,n) for various k. If one of these values is between 1 and n we have a factor of n.

FACTORING with ELLIPTIC CURVES

Basis idea: To factorize an integer n choose an elliptic curve E, a point P on E and compute, modulo n, either iP for $i = 2, 3, 4, \ldots$ or $2^j P$ for $j = 1, 2, \ldots$. The point is that in doing that one needs to compute gcd(k,n) for various k. If one of these values is between 1 and n we have a factor of n.

Factoring of large integers: The above idea can be easily parallelised and converted to using an enormous number of computers to factor a single very large n. Each computer gets some number of elliptic curves and some points on them and multiplies these points by some integers according to the rule for addition of points. If one of computers encounters, during such a computation, a need to compute $1 < gcd(k, n) < n$, factorization is finished.

FACTORING with ELLIPTIC CURVES

Basis idea: To factorize an integer n choose an elliptic curve E, a point P on E and compute, modulo n, either iP for $i = 2, 3, 4, \ldots$ or $2^j P$ for $j = 1, 2, \ldots$. The point is that in doing that one needs to compute gcd(k,n) for various k. If one of these values is between 1 and n we have a factor of n.

Factoring of large integers: The above idea can be easily parallelised and converted to using an enormous number of computers to factor a single very large n. Each computer gets some number of elliptic curves and some points on them and multiplies these points by some integers according to the rule for addition of points. If one of computers encounters, during such a computation, a need to compute $1 < gcd(k, n) < n$, factorization is finished.

Example: If curve $E : y^2 = x^3 + 4x + 4 \pmod{2773}$ and its point $P = (1, 3)$ are used, then $2P = (1771, 705)$ and in order to compute 3P one has to compute $gcd(1770, 2773) = 59$ – factorization is done.

FACTORING with ELLIPTIC CURVES

Basis idea: To factorize an integer n choose an elliptic curve E, a point P on E and compute, modulo n, either iP for $i = 2, 3, 4, \ldots$ or $2^j P$ for $j = 1, 2, \ldots$. The point is that in doing that one needs to compute gcd(k,n) for various k. If one of these values is between 1 and n we have a factor of n.

Factoring of large integers: The above idea can be easily parallelised and converted to using an enormous number of computers to factor a single very large n. Each computer gets some number of elliptic curves and some points on them and multiplies these points by some integers according to the rule for addition of points. If one of computers encounters, during such a computation, a need to compute $1 < gcd(k, n) < n$, factorization is finished.

Example: If curve $E : y^2 = x^3 + 4x + 4 \pmod{2773}$ and its point $P = (1, 3)$ are used, then $2P = (1771, 705)$ and in order to compute 3P one has to compute $gcd(1770, 2773) = 59$ – factorization is done.

Example: For elliptic curve $E : y^2 = x^3 + x - 1 \pmod{35}$ and its point $P = (1, 1)$ we have $2P = (2, 32); 4P = (25, 12); 8P = (6, 9)$ and at the attempt to compute 9P one needs to compute $gcd(15, 35) = 5$ and factorization is done.

The only things that remain to be explored is how efficient this method is and when it is more efficient than other methods.

IMPORTANT OBSERVATIONS (1)

- If $n = pq$ for primes p, q, then an elliptic curve $E \pmod n$ can be seen as a pair of elliptic curves $E \pmod p$ and $E \pmod q$.
- It follows from the Lagrange theorem that for any elliptic curve $E \pmod n$ and its point P there is an $k < n$ such that $kP = \infty$.
- In case of an elliptic curve $E \pmod p$ for some prime p, the smallest positive integer m such that $mP = \infty$ for some point P divides the number N of points on the curve $E \pmod p$. Hence $NP = \infty$.

  If N is a product of small primes, then b! will be a multiple of N for a reasonable small b. Therefore, $b!P = \infty$.
- The number with only small factors is called smooth and if all factors are smaller than an b, then it is called b-smooth.

It can be shown that the density of smooth integers is so large that if we choose a random elliptic curve $E \pmod n$ then it is a reasonable chance that n is smooth.

PRACTICALITY of FACTORING USING ECC (1)

Let us continue to discuss the following key problem for factorization using elliptic curves:

Problem: How to choose integer k such that for a given point P we should try to compute points iP or $2^i P$ for all multiples of P smaller than kP?

Idea: If one searches for m-digits factors, one chooses k in such a way that k is a multiple of as many as possible of those m-digit numbers which do not have too large prime factors. In such a case one has a good chance that k is a multiple of the number of elements of the group of points of the elliptic curve modulo n.

PRACTICALITY of FACTORING USING ECC (1)

Let us continue to discuss the following key problem for factorization using elliptic curves:

Problem: How to choose integer k such that for a given point P we should try to compute points iP or $2^i P$ for all multiples of P smaller than kP?

Idea: If one searches for m-digits factors, one chooses k in such a way that k is a multiple of as many as possible of those m-digit numbers which do not have too large prime factors. In such a case one has a good chance that k is a multiple of the number of elements of the group of points of the elliptic curve modulo n.

Method 1: One chooses an integer B and takes as k the product of all maximal powers of primes smaller than B.

PRACTICALITY of FACTORING USING ECC (1)

Let us continue to discuss the following key problem for factorization using elliptic curves:

Problem: How to choose integer k such that for a given point P we should try to compute points iP or $2^i P$ for all multiples of P smaller than kP?

Idea: If one searches for m-digits factors, one chooses k in such a way that k is a multiple of as many as possible of those m-digit numbers which do not have too large prime factors. In such a case one has a good chance that k is a multiple of the number of elements of the group of points of the elliptic curve modulo n.

Method 1: One chooses an integer B and takes as k the product of all maximal powers of primes smaller than B.

Example: In order to find a 6-digit factor one chooses B=147 and $k = 2^7 \cdot 3^4 \cdot 5^3 \cdot 7^2 \cdot 11^2 \cdot 13 \cdot \ldots \cdot 139$. The following table shows B and the number of elliptic curves one has to test:

PRACTICALITY of FACTORING USING ECC (2)

Digits of to-be-factors	6	9	12	18	24	30
B	147	682	2462	23462	162730	945922
Number of curves	10	24	55	231	833	2594

Computation time by the elliptic curves method depends on the size of factors.

ELLIPTIC CURVES FACTORIZATION - DETAILS

Given an n such that gcd(n, 6) = 1 and let the smallest factor of n be expected to be smaller than an F. One should then proceed as follows:

Choose an integer parameter r and:

1. Select, randomly, an elliptic curve

$$E : y^2 = x^3 + ax + b$$

such that $gcd(n, 4a^2 + 27b^2) = 1$ and a random point P on E.

ELLIPTIC CURVES FACTORIZATION - DETAILS

Given an n such that gcd(n, 6) = 1 and let the smallest factor of n be expected to be smaller than an F. One should then proceed as follows:

Choose an integer parameter r and:

1. Select, randomly, an elliptic curve

$$E : y^2 = x^3 + ax + b$$

   such that $gcd(n, 4a^2 + 27b^2) = 1$ and a random point P on E.

2. Choose integer bounds A,B,M such that

$$M = \prod_{j=1}^{l} p_j^{a_{p_j}}$$

   for some primes $p_1 < p_2 < \ldots < p_l \leq B$ and a_{p_j}, being the largest exponent such that $p_j^{a_j} \leq A$.

   Set j = k = 1

ELLIPTIC CURVES FACTORIZATION - DETAILS

Given an n such that gcd(n, 6) = 1 and let the smallest factor of n be expected to be smaller than an F. One should then proceed as follows:
Choose an integer parameter r and:

1. Select, randomly, an elliptic curve
$$E : y^2 = x^3 + ax + b$$
such that $gcd(n, 4a^2 + 27b^2) = 1$ and a random point P on E.
2. Choose integer bounds A,B,M such that
$$M = \prod_{j=1}^{l} p_j^{a_{p_j}}$$
for some primes $p_1 < p_2 < \ldots < p_l \leq B$ and a_{p_j}, being the largest exponent such that $p_j^{a_j} \leq A$.
Set j = k = 1
3. Calculate $p_j P$.
4. Computing gcd.
   - If $p_j P \neq O \pmod n$, then set $P = p_j P$ and reset $k \leftarrow k + 1$
     1. If $k \leq a_{p_j}$, then go to step (3).

ELLIPTIC CURVES FACTORIZATION - DETAILS II

2. If $k > a_{p_j}$, then reset $j \leftarrow j+1$, $k \leftarrow 1$.
   If $j \leq l$, then go to step (3); otherwise go to step (5)

- If $p_j P \equiv O(\mod n)$ and no factor of n was found at the computation of inverse elements, then go to step (5)

5. Reset $r \leftarrow r-1$. If $r > 0$ go to step (1); otherwise terminate with "failure".
   The "smoothness bound" B is recommended to be chosen as

$$B = e^{\sqrt{\frac{lnF(lnlnF)}{2}}}$$

and in such a case running time is

$$O(e^{\sqrt{2+o(1lnF(lnlnF))}} ln^2 n)$$

ELLIPTIC CURVES: FAQ

- How to choose (randomly) an elliptic curve E and point P on E?

ELLIPTIC CURVES: FAQ

- How to choose (randomly) an elliptic curve E and point P on E? An easy way is first choose a point $P(x, y)$ and an a and then compute $b = y^2 - x^3 - ax$ to get the curve $E : y^2 = x^3 + ax + b$.

ELLIPTIC CURVES: FAQ

- How to choose (randomly) an elliptic curve E and point P on E? An easy way is first choose a point $P(x, y)$ and an a and then compute $b = y^2 - x^3 - ax$ to get the curve $E : y^2 = x^3 + ax + b$.
- What happens at the factorization using elliptic curve method, if for a chosen curve $E \pmod{n}$ the corresponding cubic polynomial $x^3 + ax + b$ has multiple roots (that is if $4a^3 + 27b^2 = 0$) ?

ELLIPTIC CURVES: FAQ

- How to choose (randomly) an elliptic curve E and point P on E? An easy way is first choose a point $P(x, y)$ and an a and then compute $b = y^2 - x^3 - ax$ to get the curve $E : y^2 = x^3 + ax + b$.
- What happens at the factorization using elliptic curve method, if for a chosen curve $E \pmod{n}$ the corresponding cubic polynomial $x^3 + ax + b$ has multiple roots (that is if $4a^3 + 27b^2 = 0$) ? No problem, method still works.

ELLIPTIC CURVES: FAQ

- How to choose (randomly) an elliptic curve E and point P on E? An easy way is first choose a point $P(x, y)$ and an a and then compute $b = y^2 - x^3 - ax$ to get the curve $E : y^2 = x^3 + ax + b$.
- What happens at the factorization using elliptic curve method, if for a chosen curve $E \pmod n$ the corresponding cubic polynomial $x^3 + ax + b$ has multiple roots (that is if $4a^3 + 27b^2 = 0$) ? No problem, method still works.
- What kind of elliptic curves are really used in cryptography?

ELLIPTIC CURVES: FAQ

- How to choose (randomly) an elliptic curve E and point P on E? An easy way is first choose a point $P(x, y)$ and an a and then compute $b = y^2 - x^3 - ax$ to get the curve $E : y^2 = x^3 + ax + b$.
- What happens at the factorization using elliptic curve method, if for a chosen curve $E \pmod n$ the corresponding cubic polynomial $x^3 + ax + b$ has multiple roots (that is if $4a^3 + 27b^2 = 0$) ? No problem, method still works.
- What kind of elliptic curves are really used in cryptography? Elliptic curves over fields $GF(2^n)$ for $n > 150$. Dealing with such elliptic curves requires, however, slightly different rules.

FACTORIZATION

Factorization of integers is a very important problem.

FACTORIZATION

Factorization of integers is a very important problem.

A variety of techniques has been developed to deal with this problem.

FACTORIZATION

Factorization of integers is a very important problem.

A variety of techniques has been developed to deal with this problem.

So far the fastest classical factorization algorithms work in time

$$e^{O((\log n)^{\frac{1}{3}}(\log\log n)^{\frac{2}{3}})}$$

The fastest quantum algorithm for factorization works in (both quantum and classical) polynomial time.

In the rest of chapter several factorization methods will be presented and discussed.

FACTORIZATION on QUANTUM COMPUTERS

In the following we present the basic idea behind a polynomial time algorithm for quantum computers to factorize integers.

FACTORIZATION on QUANTUM COMPUTERS

In the following we present the basic idea behind a polynomial time algorithm for quantum computers to factorize integers.

Quantum computers works with superpositions of basic quantum states on which very special (unitary) operations are applied and and very special quantum features (non-locality) are used.

FACTORIZATION on QUANTUM COMPUTERS

In the following we present the basic idea behind a polynomial time algorithm for quantum computers to factorize integers.

Quantum computers works with superpositions of basic quantum states on which very special (unitary) operations are applied and and very special quantum features (non-locality) are used.

Quantum computers work not with bits, that can take on any of two values 0 and 1, but with qubits (quantum bits) that can take on any of infinitely many states $\alpha|0\rangle + \beta|1\rangle$, where α and β are complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$.

REDUCTIONS

Shor's polynomial time quantum factorization algorithm is based on an understanding that factorization problem can be reduced

1. first on the problem of solving a simple modular quadratic equation;
2. second on the problem of finding period of functions $f(x) = a^x \bmod n$.

FIRST REDUCTION

Lemma If there is a polynomial time deterministic (randomized) [quantum] algorithm to find a nontrivial solution of the modular quadratic equations

$$a^2 \equiv 1 \pmod{n},$$

then there is a polynomial time deterministic (randomized) [quantum] algorithm to factorize integers.

FIRST REDUCTION

Lemma If there is a polynomial time deterministic (randomized) [quantum] algorithm to find a nontrivial solution of the modular quadratic equations

$$a^2 \equiv 1 \pmod{n},$$

then there is a polynomial time deterministic (randomized) [quantum] algorithm to factorize integers.

Proof. Let $a \neq \pm 1$ be such that $a^2 \equiv 1 \pmod{n}$. Since

$$a^2 - 1 = (a+1)(a-1),$$

if n is not prime, then a prime factor of n has to be a prime factor of either $a+1$ or $a-1$. By using Euclid's algorithm to compute

$$gcd(a+1, n) \quad \text{and} \quad gcd(a-1, n)$$

we can find, in $O(\lg n)$ steps, a prime factor of n.

SECOND REDUCTION

The second key concept is that of the **period** of functions

$$f_{n,x}(k) = x^k \bmod n.$$

SECOND REDUCTION

The second key concept is that of the **period** of functions

$$f_{n,x}(k) = x^k \bmod n.$$

Period is the smallest integer r such that

$$f_{n,x}(k+r) = f_{n,x}(k)$$

for any k, i.e. the smallest r such that

$$x^r \equiv 1 \pmod{n}.$$

SECOND REDUCTION

The second key concept is that of the **period** of functions

$$f_{n,x}(k) = x^k \bmod n.$$

Period is the smallest integer r such that

$$f_{n,x}(k+r) = f_{n,x}(k)$$

for any k, i.e. the smallest r such that

$$x^r \equiv 1 \pmod{n}.$$

AN ALGORITHM TO SOLVE EQUATION $x^2 \equiv 1 \pmod{n}$.

1. *Choose randomly $1 < a < n$.*
2. *Compute $\gcd(a, n)$. If $\gcd(a, n) \neq 1$ we have a factor.*
3. *Find period r of function $a^k \bmod n$.*
4. *If r is odd or $a^{r/2} \equiv \pm 1 \pmod{n}$, then go to step 1; otherwise stop.*

If this algorithm stops, then $a^{r/2}$ is a non-trivial solution of the equation

$$x^2 \equiv 1 \pmod{n}.$$

EXAMPLE

Let $n = 15$. Select $a < 15$ such that $gcd(a, 15) = 1$.
$\{$The set of such a is $\{2, 4, 7, 8, 11, 13, 14\}\}$

Choose $a = 11$. Values of 11^x mod 15 are then

$$11, 1, 11, 1, 11, 1$$

whiach gives $r = 2$.

Hence $a^{r/2} = 11 \pmod{15}$. Therefore

$$gcd(15, 12) = 3, \qquad gcd(15, 10) = 5$$

EXAMPLE

Let $n = 15$. Select $a < 15$ such that $gcd(a, 15) = 1$.
$\{$The set of such a is $\{2, 4, 7, 8, 11, 13, 14\}\}$

Choose $a = 11$. Values of 11^x mod 15 are then

$$11, 1, 11, 1, 11, 1$$

whiach gives $r = 2$.

Hence $a^{r/2} = 11 \pmod{15}$. Therefore

$$gcd(15, 12) = 3, \qquad gcd(15, 10) = 5$$

For $a = 14$ we get again $r = 2$, but in this case

$$14^{2/2} \equiv -1 \pmod{15}$$

and the following algorithm fails.

1. *Choose randomly* $1 < a < n$.
2. *Compute* $gcd(a, n)$. *If* $gcd(a, n) \neq 1$ *we have a factor.*
3. *Find period* r *of function* $a^k \bmod n$.
4. *If* r *is odd or* $a^{r/2} \equiv \pm 1 \pmod{n}$,*then go to step 1; otherwise stop.*

EFFICIENCY of REDUCTION

Lemma If $1 < a < n$ satisfying $gcd(n, a) = 1$ is selected in the above algorithm randomly and n is not a power of prime, then

$$Pr\{r \text{ is even and } a^{r/2} \not\equiv \pm 1\} \geq \frac{9}{16}.$$

1. *Choose randomly* $1 < a < n$.
2. *Compute* $gcd(a, n)$. *If* $gcd(a, n) \neq 1$ *we have a factor.*
3. *Find period* r *of function* $a^k \bmod n$.
4. *If* r *is odd or* $a^{r/2} \equiv \pm 1 \pmod n$,*then go to step 1; otherwise stop.*

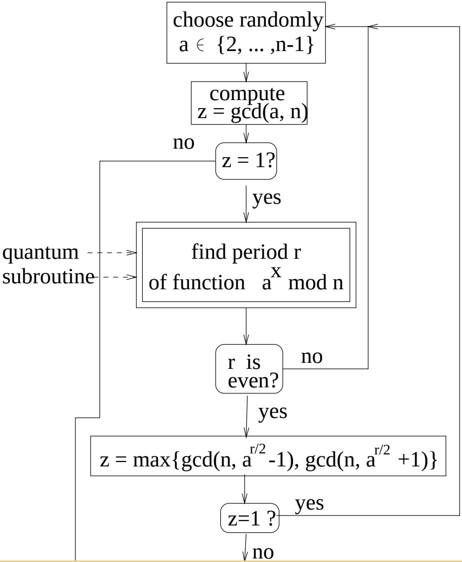
Corollary If there is a polynomial time randomized [quantum] algorithm to compute the period of the function

$$f_{n,a}(k) = a^k \bmod n,$$

then there is a polynomial time randomized [quantum] algorithm to find non-trivial solution of the equation $a^2 \equiv 1 \pmod n$ (and therefore also to factorize integers).

A GENERAL SCHEME for Shor's ALGORITHM

The following flow diagram shows the general scheme of Shor's quantum factorization algorithm



Fermat FACTORIZATION METHOD

Factorization of so-called Fermat numbers $2^{2^i}+1$ is a good example to illustrate progress that has been made in the area of factorization.

Pierre de Fermat (1601-65) expected that all numbers

$$F_i = 2^{2^i} + 1 \qquad i \geq 1$$

are primes.

This is true for $i = 1, \ldots, 4$. $F_1 = 5$, $F_2 = 17$, $F_3 = 257$, $F_4 = 65537$.

1732 L. Euler found that $F_5 = 4294967297 = 641 \cdot 6700417$

Fermat FACTORIZATION METHOD

Factorization of so-called Fermat numbers $2^{2^i}+1$ is a good example to illustrate progress that has been made in the area of factorization.
Pierre de Fermat (1601-65) expected that all numbers

$$F_i = 2^{2^i}+1 \qquad i \geq 1$$

are primes.

This is true for $i = 1, \ldots, 4$. $F_1 = 5$, $F_2 = 17$, $F_3 = 257$, $F_4 = 65537$.

1732 L. Euler found that $F_5 = 4294967297 = 641 \cdot 6700417$
1880 Landry+LeLasser found that

$$F_6 = 18446744073709551617 = 274177 \cdot 67280421310721$$

Fermat FACTORIZATION METHOD

Factorization of so-called Fermat numbers $2^{2^i}+1$ is a good example to illustrate progress that has been made in the area of factorization.

Pierre de Fermat (1601-65) expected that all numbers

$$F_i = 2^{2^i} + 1 \qquad i \geq 1$$

are primes.

This is true for $i = 1, \ldots, 4$. $F_1 = 5$, $F_2 = 17$, $F_3 = 257$, $F_4 = 65537$.

1732 L. Euler found that $F_5 = 4294967297 = 641 \cdot 6700417$

1880 Landry+LeLasser found that

$$F_6 = 18446744073709551617 = 274177 \cdot 67280421310721$$

1970 Morrison+Brillhart found factorization for $F_7 = (39 digits)$

$$F_7 = 340282366920938463463374607431768211457 =$$
$$= 5704689200685129054721 \cdot 5964958912749721 7$$

Fermat FACTORIZATION METHOD

Factorization of so-called Fermat numbers $2^{2^i}+1$ is a good example to illustrate progress that has been made in the area of factorization.

Pierre de Fermat (1601-65) expected that all numbers

$$F_i = 2^{2^i}+1 \qquad i \geq 1$$

are primes.

This is true for $i = 1, \ldots, 4$. $F_1 = 5$, $F_2 = 17$, $F_3 = 257$, $F_4 = 65537$.

1732 L. Euler found that $F_5 = 4294967297 = 641 \cdot 6700417$

1880 Landry+LeLasser found that

$$F_6 = 18446744073709551617 = 274177 \cdot 67280421310721$$

1970 Morrison+Brillhart found factorization for $F_7 = (39 digits)$

$$F_7 = 340282366920938463463374607431768211457 =$$
$$= 5704689200685129054721 \cdot 5964958912749721 7$$

1980 Brent+Pollard found factorization for F_8

Fermat FACTORIZATION METHOD

Factorization of so-called Fermat numbers $2^{2^i}+1$ is a good example to illustrate progress that has been made in the area of factorization.
Pierre de Fermat (1601-65) expected that all numbers

$$F_i = 2^{2^i}+1 \qquad i \geq 1$$

are primes.

This is true for $i = 1, \ldots, 4$. $F_1 = 5$, $F_2 = 17$, $F_3 = 257$, $F_4 = 65537$.

1732 L. Euler found that $F_5 = 4294967297 = 641 \cdot 6700417$
1880 Landry+LeLasser found that

$$F_6 = 18446744073709551617 = 274177 \cdot 67280421310721$$

1970 Morrison+Brillhart found factorization for $F_7 = (39 digits)$

$$F_7 = 340282366920938463463374607431768211457 =$$
$$= 5704689200685129054721 \cdot 5964958912749721 7$$

1980 Brent+Pollard found factorization for F_8
1990 A. K. Lenstra+ ... found factorization for F_9 (155 digits)

Fermat TEST

It follows from the Little Fermat Theorem that if p is a prime, then for all $0 < b < p$, we have

$$b^{p-1} \equiv 1 \pmod{p}$$

Can we say that n is prime if and only if for all $0 < b < n$, we have

$$b^{n-1} \equiv 1 \pmod{n}?$$

Fermat TEST

It follows from the Little Fermat Theorem that if p is a prime, then for all $0 < b < p$, we have

$$b^{p-1} \equiv I \pmod{p}$$

Can we say that n is prime if and only if for all $0 < b < n$, we have

$$b^{n-1} \equiv I \pmod{n}?$$

No, there are composed numbers n, so-called Carmichael numbers, such that for all $0 < b < n$ that are co-prime with n it holds

$$b^{n-1} \equiv I \pmod{n}$$

Such number is, for example, n=561.

POLLARD ρ-METHOD

A variety of factorization algorithms, of complexity around $O(\sqrt{p})$ where p is the smallest prime factor of n, is based on the following idea:

- A function f is taken that "behaves like a randomizing function" and $f(x) \equiv f(x \bmod p) \pmod p$ for any factor p of n – usually $f(x) = x^2 + 1$
- A random x_0 is taken and iteration

$$x_{i+1} = f(x_i) \mod n$$

  is performed (this modulo n computation actually "hides" modulo p computation in the following sense: if $x'_0 = x_0$, $x'_{i+1} = f(x'_i) \mod n$, then $x'_i = x_i \mod p$)
- Since $\mathbf{Z}_p$ is finite, the shape of the sequence x'_i will remind the letter ρ, with a tail and a loop. Since f is "random", the loop modulo n rarely synchronizes with the loop modulo p
- The loop is easy to detect by GCD-computations and it can be shown that the total length of tail and loop is $O(\sqrt{p})$.

LOOP DETECTION

In order to detect the loop it is enough to perform the following computation:

$a \leftarrow x_0$; $b \leftarrow x_0$;
repeat
  $a \leftarrow f(a)$;
  $b \leftarrow f(f(b))$;
until a = b

Iteration ends if $a_t = b_{2t}$ for some t greater than the tail length and a multiple of the loop length.

FIRST Pollard ρ-ALGORITHM

Input: an integer n with a factor smaller than B
Complexity: $O(\sqrt{B})$ of arithmetic operations

$x_0 \leftarrow random$; $a \leftarrow x_0$; $b \leftarrow x_0$;
do
  $a \leftarrow f(a) \mod n$;
  $b \leftarrow f(f(b) \mod n) \mod n$;
until gcd(a - b, n) $\neq 1$
output gcd(a - b, n)

The proof that complexity of the first Pollard factorization ρ-algorithm is given by $O(N^{\frac{1}{4}})$ arithmetic operations is based on the following result:
Lemma Let x_0 be random and f be "random" in Z_p, $x_{i+1} = f(x_i)$. The probability that all elements of the sequence

$$x_0, x_1, \ldots, x_t$$

are pairwise different when $t = 1 + \lfloor (2\lambda p)^{\frac{1}{2}} \rfloor$ is less than $e^{-\lambda}$.

SECOND Pollard ρ-ALGORITHM

Basic idea

1. Choose an easy to compute $f : Z_n \to Z_n$ and $x_0 \in Z_n$.

   Example $f(x) = x^2 + 1$

2. Keep computing $x_{i+1} = f(x_j)$, $j = 0, 1, 2, \dots$ and $gcd(x_j - x_k, n)$, $k \leq j$. (Observe that if $x_j \equiv x_k \mod p$ for a prime factor p of n, then $gcd(x_j - x_k, n) \leq p$.)

   Example n $= 91$, $f(x) = x^2 + 1$, $x_0 = 1$, $x_1 = 2$, $x_2 = 5$, $x_3 = 26$

$$gcd(x_3 - x_2, n) = gcd(26 - 5, 91) = 7$$

Remark: In the ρ-method, it is important to choose a function f in such a way that f maps Z_n into Z_n in a "random" way.

Basic question: How good is the ρ-method?
(How long we expect to have to wait before we get two values x_j, x_k such that $gcd(x_j - x_k, n) \neq 1$, if n is not a prime?)

ρ-ALGORITHM

A simplification of the basic idea: For each k compute $gcd(x_k - x_j, n)$ for just one $j < k$.

Choose $f : Z_n \rightarrow Z_n, x_0$, compute $x_k = f(x_{k-1}), k > 0$.
If k is an (h +1)-bit integer, i.e. $2^h \leq k \leq 2^{h+1}$, then compute $gcd(x_k, x_{2^h-1})$.

A simplification of the basic idea: For each k compute $gcd(x_k - x_j, n)$ for just one $j < k$.

Choose $f : Z_n \rightarrow Z_n, x_0$, compute $x_k = f(x_{k-1}), k > 0$.
If k is an (h +1)-bit integer, i.e. $2^h \leq k \leq 2^{h+1}$, then compute $gcd(x_k, x_{2^h - 1})$.

Example n = 4087, $f(x) = x^2 + x + 1, x_0 = 2$

x_1 = f(2) = 7,	gcd($x_1 - x_0$, n) = 1
x_2 = f(7) = 57,	gcd($x_2 - x_1$, n) = gcd(57 – 7, n) = 1
x_3 = f(57) = 3307,	gcd($x_3 - x_1$, n) = gcd(3307 - 7, n) = 1
x_4 = f(3307) = 2745,	gcd($x_4 - x_3$, n) = gcd(2745 - 3307, n) = 1
x_5 = f(2746) = 1343,	gcd($x_5 - x_3$, n) = gcd(1343 - 3307, n) = 1
x_6 = f(1343) = 2626,	gcd($x_6 - x_3$, n) = gcd(2626 - 3307, n) = 1
x_7 = f(2626) = 3734,	gcd($x_7 - x_3$, n) = gcd(3734 - 3307, n) = 61

A simplification of the basic idea: For each k compute $gcd(x_k - x_j, n)$ for just one $j < k$.

Choose $f : Z_n \rightarrow Z_n, x_0$, compute $x_k = f(x_{k-1}), k > 0$.
If k is an (h +1)-bit integer, i.e. $2^h \leq k \leq 2^{h+1}$, then compute $gcd(x_k, x_{2^h-1})$.

Example n = 4087, $f(x) = x^2 + x + 1, x_0 = 2$

x_1 = f(2) = 7,	gcd($x_1 - x_0$, n) = 1
x_2 = f(7) = 57,	gcd($x_2 - x_1$, n) = gcd(57 – 7, n) = 1
x_3 = f(57) = 3307,	gcd($x_3 - x_1$, n) = gcd(3307 - 7, n) = 1
x_4 = f(3307) = 2745,	gcd($x_4 - x_3$, n) = gcd(2745 - 3307, n) = 1
x_5 = f(2746) = 1343,	gcd($x_5 - x_3$, n) = gcd(1343 - 3307, n) = 1
x_6 = f(1343) = 2626,	gcd($x_6 - x_3$, n) = gcd(2626 - 3307, n) = 1
x_7 = f(2626) = 3734,	gcd($x_7 - x_3$, n) = gcd(3734 - 3307, n) = 61

Disadvantage We likely will not detect the first case such that for some k_0 there is a $j_0 < k_0$ such that $gcd(x_{k0} - x_{j0}, n) > 1$.
This is no real problem! Let k_0 have h + 1 bits. Set $j = 2^{h+1} - 1, k = j + k_0 - j_0$. k has (h+2) bits, $gcd(x_k - x_j, n) > 1$

$$k < 2^{h+2} = 4 \cdot 2^h \leq 4k_0.$$

ρ-ALGORITHM

Theorem Let n be odd and composite and $1 < r < \sqrt{n}$ its factor. If f, x_0 are chosen randomly, then ρ algorithm reveals r in $O(\sqrt[4]{n} log^3 n)$ bit operations with high probability. More precisely, there is a constant $C > 0$ such that for any $\lambda > 0$, the probability that the ρ algorithm fails to find a nontrivial factor of n in $C\sqrt{\lambda}\sqrt[4]{n} log^3 n$ bit operations is less than $e^{-\lambda}$.

ρ-ALGORITHM

Theorem Let n be odd and composite and $1 < r < \sqrt{n}$ its factor. If f, x_0 are chosen randomly, then ρ algorithm reveals r in $O(\sqrt[4]{n}log^3 n)$ bit operations with high probability. More precisely, there is a constant $C > 0$ such that for any $\lambda > 0$, the probability that the ρ algorithm fails to find a nontrivial factor of n in $C\sqrt{\lambda}\sqrt[4]{n}log^3 n$ bit operations is less than $e^{-\lambda}$.

Proof Let C_1 be a constant such that gcd(y - z, n) can be computed in $C_1 log^3 n$ bit operations whenever $y, z < n$.
Let C_2 be a constant such that f(x) mod n can be computed in $C_2 log^2 n$ bit operations if $x < n$.
If k_0 is the first index for which there exists $j_0 < k_0$ with $x_{k0} \equiv x_{j0} \mod r$, then the ρ-algorithm finds r in $k \leq 4k_0$ steps.
The total number of bit operations is bounded by $\rightarrow 4k_0(C_1 log^3 n + C_2 log^2 n)$
By Lemma the probability that k_0 is greater than $1 + \sqrt{2\lambda r}$ is less than $e^{-\lambda}$.
If $k_0 \leq 1 + \sqrt{2\lambda r}$, then the number of bit operations needed to find r is bounded by

$$4(1 + \sqrt{2\lambda r})(C_1 log^3 n - C_2 log^2 n) < 4(1 + \sqrt{2\lambda}\sqrt[4]{n})(C_1 log^3 n + C_2 log^2 n)$$

If we choose $C > 4\sqrt{2}(C_1 + C_2)$, then we have that r will be found in $C\sqrt{\lambda}\sqrt[4]{n}log^3 n$ bit operations – unless we made uniform choice of (f, x_0) the probability of which a is at most $e^{-\lambda}$.

COMMENTS

Pollard ρ-method works fine for integers n with a small factor.

Next method, so called Pollard (p-1)-method, works fine for n having a prime factor p such that all prime factors of p-1 are small.

When all prime factors of p-1 are smaller than a B, we say that p-1 is B-smooth.

POLLARD's p-1 algorithm

Pollard's algorithm (to factor n given a bound b on factors).

a := 2;

for j=2 **to** b **do** $a := a^j \mod n$;

$f := gcd(a-1, n)$; $\qquad f = gcd(2^{b!} - 1, n)$

if $1 < f < n$ **then** f is a factor of n **otherwise** failure

Indeed, let p be a prime divisor of n and $q < b$ for every prime $q|(p-1)$.
(Hence $(p-1)|b!$).

At the end of the **for**-loop we have

$$a \equiv 2^{b!} \pmod{n}$$

and therefore

$$a \equiv 2^{b!} \pmod{p}$$

By Fermat theorem $2^{p-1} \equiv 1 \pmod{p}$ and since $(p-1)|b!$ we get $a \equiv 2^{b!} \equiv 1 \pmod{p}$.
and therefore we have $p|(a-1)$
Hence

$$p|gcd(a-1, n)$$

IMPORTANT OBSERVATIONS II

Pollard ρ-method works fine for numbers with a small factor.

The p-1 method requires that p-1 is smooth. The elliptic curve method requires only that there are enough smooth integers near p and so at least one of randomly chosen integers near p is smooth.

This means that the elliptic curves factorization method succeeds much more often than p-1 method.

Fermat factorization and Quadratic Sieve method discussed later works fine if integer has two factors of almost the same size.

Fermat FACTORIZATION I

If n = pq, $p < \sqrt{n}$, then

$$n = \left(\frac{q+p}{2}\right)^2 - \left(\frac{q-p}{2}\right)^2 = a^2 - b^2$$

Therefore, in order to find a factor of n, we need only to investigate the values

$$x = a^2 - n$$

$$\text{for } a = \left\lceil\sqrt{n}\right\rceil + 1, \left\lceil\sqrt{n}\right\rceil + 2, \ldots, \frac{(n-1)}{2}$$

until a perfect square is found.

Fermat FACTORIZATION

Basic idea: Factorization is easy if one finds x, y such that $n|(x^2 - y^2)$

Proof: If n divides (x + y)(x - y) and n does not divide neither x+y nor x-y, then one factor of n has to divide x+y and another one x-y.

Example $n = 7429 = 227^2 - 210^2$, $x = 227$, $y = 210$
$x - y = 17$ $x + y = 437$
$\gcd(17, 7429) = 17$ $\gcd(437, 7429) = 437$.

How to find such x and y?

First idea: one tries all t starting with $\sqrt{n}$ until $t^2 - n$ is a square S^2.

Second idea: One forms a system of (modular) linear equations and determines x and y from the solutions of such a system.

number of digits of n	50	60	70	80	90	100	110	120
number of equations	3000	4000	7400	15000	30000	51000	120000	245000

METHOD of QUADRATIC SIEVE to FACTORIZE an INTEGER n

Step 1 One finds numbers x such that $x^2 - n$ is small and has small factors.

Example
$$\left.\begin{array}{l} 83^2 - 7429 = -540 = (-1)\cdot 2^2\cdot 3^3\cdot 5 \\ 87^2 - 7429 = 140 = 2^2 \cdot 5 \cdot 7 \\ 88^2 - 7429 = 315 = 3^2 \cdot 5 \cdot 7 \end{array}\right\} \text{relations}$$

METHOD of QUADRATIC SIEVE to FACTORIZE an INTEGER n

Step 1 One finds numbers x such that $x^2 - n$ is small and has small factors.

Example
$$\left.\begin{array}{l} 83^2 - 7429 = -540 = (-1)\cdot 2^2 \cdot 3^3 \cdot 5 \\ 87^2 - 7429 = 140 = 2^2 \cdot 5 \cdot 7 \\ 88^2 - 7429 = 315 = 3^2 \cdot 5 \cdot 7 \end{array}\right\} \text{relations}$$

Step 2 One multiplies some of the relations if their product is a square.
For example

$$(87^2 - 7429)(88^2 - 7429) = 2^2 \cdot 3^2 \cdot 5^2 \cdot 7^2 = 210^2$$

Now

$$(87 \cdot 88)^2 \equiv (87^2 - 7429)(88^2 - 7429) \mod 7429$$
$$227^2 \equiv 210^2 \mod 7429$$

Hence 7429 divides $227^2 - 210^2$.
Formation of equations: For the i-th relation one takes a variable λ_i and forms the expression

$$((-1)\cdot 2^2 \cdot 3^3 \cdot 5)^{\lambda_1} \cdot (2^2 \cdot 5 \cdot 7)^{\lambda_2} \cdot (3^2 \cdot 5 \cdot 7)^{\lambda_3} = (-1)^{\lambda_1} \cdot 2^{2\lambda_1 + 2\lambda_2} \cdot 3^{2\lambda_1 + 2\lambda_2} \cdot 5^{\lambda_1 + \lambda_2 + \lambda_3} \cdot 7^{\lambda_2 + \lambda_3}$$

If this is to form a square the following equations have to hold

$$\begin{aligned} \lambda_1 &\equiv 0 \mod 2 \\ \lambda_1 + \lambda_2 + \lambda_3 &\equiv 0 \mod 2 \\ \lambda_2 + \lambda_3 &\equiv 0 \mod 2 \\ \lambda_1 = 0, \quad & \lambda_2 = \lambda_3 = 1 \end{aligned}$$

METHOD of QUADRATIC SIEVE to FACTORIZE n

Problem How to find relations?

Using the algorithm called Quadratic sieve method.

Problem How to find relations?

Using the algorithm called Quadratic sieve method.

Step 1 One chooses a set of primes that can be factors – a so-called factor basis.

One chooses an m such that $m^2 - n$ is small and considers numbers $(m+u)^2 - n$ for $-k \le u \le k$ for small k.

One then tries to factor all $(m+u)^2 - n$ with primes from the factor basis, from the smallest to the largest.

u $(m+u)^2 - n$	-3 -540	-2 -373	-1 -204	0 -33	1 140	2 315	3 492
Sieve with 2	-135		-51		35		123
Sieve with 3	-5		-17	-11		35	41
Sieve with 5	-1				7	7	
Sieve with 7					1	1	

In order to factor a 129-digit number from the RSA challenge they used

8 424 486 relations

569 466 equations

544 939 elements in the factor base

APPENDIX to CHAPTER 8

HISTORY of ELLIPTIC CURVES CRYPTOGRAPHY

- The use of elliptic curves in cryptography was suggested independently by Neal Koblitz and Victor S. Miller in 1985.
- Behind this method is a believe that the discrete logarithm of a random elliptic curve element with respect to publicly known base point is infeasible.
- At first Elliptic curves over a prime finite field were used for ECC. Later also elliptic curves over the fields $GF(2^m)$ started to be used.
- In 2005 the US NSA endorsed to use ECC (Elliptic curves cryptography) with 384-bit key to protect information classified as "top secret".
- There are patents in force covering certain aspects of ECC technology.
- Elliptic curves have been first used for factorization by Lenstra.
- Elliptic curves played an important role in perhaps most celebrated mathematical proof of the last hundred years - in the proof of Fermat's Last Theorem - due to A. Wiles and R. Taylor.

SECURITY of ELLIPTIC CURVE CRYPTOGRAPHY

- Security of ECC depends on the difficulty of solving the discrete logarithm problem over elliptic curves.
- Two general methods of solving such discrete logarithm problems are known.
- The square root method and Silver-Pohling-Hellman (SPH) method.
- SPH method factors the order of a curve into small primes and solves the discrete logarithm problem as a combination of discrete logarithms for small numbers.
- Computation time of the square root method is proportional to $O(\sqrt{e^n})$ where n is the order of the based element of the curve.

FACTORIZATION of a 512-BIT NUMBER

On August 22, 1999, a team of scientists from 6 countries found, after 7 months of computing, using 300 very fast SGI and SUN workstations and Pentium II, factors of the so-called RSA-155 number with 512 bits (about 155 digits).

FACTORIZATION of a 512-BIT NUMBER

On August 22, 1999, a team of scientists from 6 countries found, after 7 months of computing, using 300 very fast SGI and SUN workstations and Pentium II, factors of the so-called RSA-155 number with 512 bits (about 155 digits).

RSA-155 was a number from a Challenge list issue by the US company RSA Data Security and "represented" 95 % of 512-bit numbers used as the key to protect electronic commerce and financial transmissions on Internet.

Factorization of RSA-155 would require in total 37 years of computing time on a single computer.

When in 1977 Rivest and his colleagues challenged the world to factor RSA-129, he estimated that, using knowledge of that time, factorization of RSA-129 would require 10^{16} years.

LARGE NUMBERS

Hindus named many large numbers – one having 153 digits.

Romans initially had no terms for numbers larger than 10^4.

Greeks had a popular belief that no number is larger than the total count of sand grains needed to fill the universe.

Large numbers with special names:

googol - 10^{100} googolplex - $10^{10^{100}}$

Hindus named many large numbers – one having 153 digits.

Romans initially had no terms for numbers larger than 10^4.

Greeks had a popular belief that no number is larger than the total count of sand grains needed to fill the universe.

Large numbers with special names:

googol - 10^{100} googolplex - $10^{10^{100}}$

FACTORIZATION of very large NUMBERS

W. Keller factorized F_{23471} which has 10^{7000} digits.

J. Harley factorized: $10^{10^{1000}} + 1$.

One factor: 316,912,650,057,350,374,175,801,344,000,001

1992 E. Crandal, Doenias proved, using a computer that F_{22}, which has more than million of digits, is composite (but no factor of F_{22} is known).

Number $10^{10^{10^{34}}}$ was used to develop a theory of the distribution of prime numbers.

Part IX

Identification, authentication, secret sharing and e-commerce

USER IDENTIFICATION and MESSAGE AUTHENTICATION, SECRET SHARING and E-COMMERCE

Most of today's cryptographic applications ask for authenticity of data rather than for secret data.

Main related problems to deal with are:

1. User identification (authentication): How can a person/computer prove her/his identity?

USER IDENTIFICATION and MESSAGE AUTHENTICATION, SECRET SHARING and E-COMMERCE

Most of today's cryptographic applications ask for authenticity of data rather than for secret data.

Main related problems to deal with are:

1. User identification (authentication): How can a person/computer prove her/his identity?
2. Message authentication: Can tools be provided to find out, for the recipient, that the message is indeed from the person who was supposed to send it?

USER IDENTIFICATION and MESSAGE AUTHENTICATION, SECRET SHARING and E-COMMERCE

Most of today's cryptographic applications ask for authenticity of data rather than for secret data.

Main related problems to deal with are:

1. User identification (authentication): How can a person/computer prove her/his identity?
2. Message authentication: Can tools be provided to find out, for the recipient, that the message is indeed from the person who was supposed to send it?
3. Message integrity (authentication): Can tools be provided to decide for the recipient whether or not the message was changed on the fly?

USER IDENTIFICATION and MESSAGE AUTHENTICATION, SECRET SHARING and E-COMMERCE

Most of today's cryptographic applications ask for authenticity of data rather than for secret data.

Main related problems to deal with are:

1. User identification (authentication): How can a person/computer prove her/his identity?
2. Message authentication: Can tools be provided to find out, for the recipient, that the message is indeed from the person who was supposed to send it?
3. Message integrity (authentication): Can tools be provided to decide for the recipient whether or not the message was changed on the fly?

Important practical objectives are to find identification schemes that are so simple that they can be implemented on smart cards – they are essentially credit cards equipped with a chip that can perform arithmetical operations and communications.

USER IDENTIFICATION and MESSAGE AUTHENTICATION, SECRET SHARING and E-COMMERCE

Most of today's cryptographic applications ask for authenticity of data rather than for secret data.

Main related problems to deal with are:

1. User identification (authentication): How can a person/computer prove her/his identity?
2. Message authentication: Can tools be provided to find out, for the recipient, that the message is indeed from the person who was supposed to send it?
3. Message integrity (authentication): Can tools be provided to decide for the recipient whether or not the message was changed on the fly?

Important practical objectives are to find identification schemes that are so simple that they can be implemented on smart cards – they are essentially credit cards equipped with a chip that can perform arithmetical operations and communications.

Secret sharing among a group of users so only well specify subsets of them can discover it is another often used cryptographic primitive we will deal with

USER IDENTIFICATION and MESSAGE AUTHENTICATION, SECRET SHARING and E-COMMERCE

Most of today's cryptographic applications ask for authenticity of data rather than for secret data.

Main related problems to deal with are:

1. User identification (authentication): How can a person/computer prove her/his identity?
2. Message authentication: Can tools be provided to find out, for the recipient, that the message is indeed from the person who was supposed to send it?
3. Message integrity (authentication): Can tools be provided to decide for the recipient whether or not the message was changed on the fly?

Important practical objectives are to find identification schemes that are so simple that they can be implemented on smart cards – they are essentially credit cards equipped with a chip that can perform arithmetical operations and communications.

Secret sharing among a group of users so only well specify subsets of them can discover it is another often used cryptographic primitive we will deal with

E-commerce: One of the main new applications of the cryptographic techniques is to establish secure and convenient manipulation with digital money (e-money), especially for e-commerce.

USER IDENTIFICATION (AUTHENTICATION)

User identification (authentication) is a process at which one party (often referred to as a Prover or Alice) convinces a second party (often referred to as a Verifier or Bob) of Prover's identity.

USER IDENTIFICATION (AUTHENTICATION)

User identification (authentication) is a process at which one party (often referred to as a Prover or Alice) convinces a second party (often referred to as a Verifier or Bob) of Prover's identity.

Namely, that the Prover (Alice) herself has actually participated in the identification process.

USER IDENTIFICATION (AUTHENTICATION)

User identification (authentication) is a process at which one party (often referred to as a Prover or Alice) convinces a second party (often referred to as a Verifier or Bob) of Prover's identity.

Namely, that the Prover (Alice) herself has actually participated in the identification process. In other words that the Prover has been herself active in proving her identity in the time the confirmative evidence of her identity has been required).

USER IDENTIFICATION (AUTHENTICATION)

User identification (authentication) is a process at which one party (often referred to as a Prover or Alice) convinces a second party (often referred to as a Verifier or Bob) of Prover's identity.

Namely, that the Prover (Alice) herself has actually participated in the identification process. In other words that the Prover has been herself active in proving her identity in the time the confirmative evidence of her identity has been required).

The purpose of any identification (authentication) process is to preclude (vylucit) some impersonation (zosobnenie) of one person (the Prover) by someone else.

Identification usually serves to control access to a resource (often a resource should be accessed only by privileged users).

OBJECTIVES of IDENTIFICATIONS

User identification process has to satisfy the following objectives:

- The Verifier has to accept Prover's identity if both parties are honest;

OBJECTIVES of IDENTIFICATIONS

User identification process has to satisfy the following objectives:

- The Verifier has to accept Prover's identity if both parties are honest;
- The Verifier cannot later, after a successful identification, act as the Prover and identify himself (as the Prover) to another Verifier;

OBJECTIVES of IDENTIFICATIONS

User identification process has to satisfy the following objectives:

- The Verifier has to accept Prover's identity if both parties are honest;
- The Verifier cannot later, after a successful identification, act as the Prover and identify himself (as the Prover) to another Verifier;
- A dishonest party, say E, that would claim to be the other party, say A, has only negligible chance to identify itself successfully as A;

OBJECTIVES of IDENTIFICATIONS

User identification process has to satisfy the following objectives:

- The Verifier has to accept Prover's identity if both parties are honest;
- The Verifier cannot later, after a successful identification, act as the Prover and identify himself (as the Prover) to another Verifier;
- A dishonest party, say E, that would claim to be the other party, say A, has only negligible chance to identify itself successfully as A;
- Each of the above conditions remains true even if an attacker has observed, or has participated in, several identification processes of the same party.

Identification protocols have to satisfy two security conditions:

1. If one party, say Bob (a Verifier), gets a message from the other party, that claims to be Alice (a Prover), then Bob is able to verify that the sender was indeed Alice.
2. There is no way to pretend, for a third party, say Charles, when communicating with Bob, that he is Alice without Bob having a large chance to find that out.

IDENTIFICATION SYSTEM BASED on a PKC

- Alice chooses a random r and sends $e_B(r)$ to Bob.
- Alice identifies a communicating person as Bob if he can send her back r.
- Bob identifies a communicating person as Alice if she can send him back r.

IDENTIFICATION SYSTEM BASED on a PKC

- Alice chooses a random r and sends $e_B(r)$ to Bob.
- Alice identifies a communicating person as Bob if he can send her back r.
- Bob identifies a communicating person as Alice if she can send him back r.

A misuse of the above system

We show that (any non-honest) Alice could misuse the above identification scheme.

Indeed, Alice could intercept a communication of Jane (some new "player") with Bob, and get a cryptotext $e_B(w)$, the one Jana has been sending to Bob, and then Alice could send $e_B(w)$ to Bob.

Honest Bob, who follows fully the protocol, would then return w to Alice and she would get this way the plaintext w.

IDENTIFICATION SYSTEM BASED on a PKC - a better version

- Alice chooses a random r and sends $e_B(r)$ to Bob.
- Alice identifies a communicating person as Bob if he can send her back r through $e_A(r, r_1)$ for a random r_1.
- Bob identifies a communicating person as Alice if she can send him back r, r_1.

USER IDENTIFICATION

Static means of identification: People can be identified by their (a) attributes (fingerprints), possessions (passports), or knowledge.

USER IDENTIFICATION

Static means of identification: People can be identified by their (a) attributes (fingerprints), possessions (passports), or knowledge.

Dynamic means of identification: Challenge and respond protocols.

USER IDENTIFICATION

Static means of identification: People can be identified by their (a) attributes (fingerprints), possessions (passports), or knowledge.

Dynamic means of identification: Challenge and respond protocols.

Example: Both Alice and Bob share a key k and a one-way function f_k.

1. Bob sends Alice a random number, or a random string, RAND.
2. Alice sends Bob $PI = f_k(RAND)$.
3. If Bob gets PI, then he verifies whether $PI = f_k(RAND)$.

If yes, he starts to believe that the person he has communicated with is Alice (more exactly that it is the person who sent RAND to him.

USER IDENTIFICATION

Static means of identification: People can be identified by their (a) attributes (fingerprints), possessions (passports), or knowledge.

Dynamic means of identification: Challenge and respond protocols.

Example: Both Alice and Bob share a key k and a one-way function f_k.

1. Bob sends Alice a random number, or a random string, RAND.
2. Alice sends Bob $PI = f_k(RAND)$.
3. If Bob gets PI, then he verifies whether $PI = f_k(RAND)$.

If yes, he starts to believe that the person he has communicated with is Alice (more exactly that it is the person who sent RAND to him.

The process can be repeated to increase probability of a correct identification.

USER IDENTIFICATION

Static means of identification: People can be identified by their (a) attributes (fingerprints), possessions (passports), or knowledge.

Dynamic means of identification: Challenge and respond protocols.

Example: Both Alice and Bob share a key k and a one-way function f_k.

1. Bob sends Alice a random number, or a random string, RAND.
2. Alice sends Bob $PI = f_k(RAND)$.
3. If Bob gets PI, then he verifies whether $PI = f_k(RAND)$.

If yes, he starts to believe that the person he has communicated with is Alice (more exactly that it is the person who sent RAND to him.

The process can be repeated to increase probability of a correct identification.

MESSAGE AUTHENTICATION – to be discussed in details later

MAC -method (Message Authentication Code) Alice and Bob share a key k and a encoding algorithm A_k

1. With a message m, Alice sends (m, A_k (m)) – MAC is here $A_k(m)$
2. If Bob gets (m', MAC), then he computes A_k (m') and compares it with MAC.

THREE-WAY AUTHENTICATION and also KEY-AGREEMENT I

A PKC will be used with encryption/decryption algorithms (e_U, d_U), for each user U, and DSS with signing/verification algorithms(s_U, v_U). Alice and Bob will have their, public, identity strings I_A and I_B.

A PKC will be used with encryption/decryption algorithms (e_U, d_U), for each user U, and DSS with signing/verification algorithms(s_U, v_U). Alice and Bob will have their, public, identity strings I_A and I_B.

1. Alice chooses a random integer r_A, sets $\mathrm{t} = (I_B,\ r_A)$, signs it as $sig_{s_A}(t)$ and sends $m_1 = (t, sig_{s_A}(t))$ to Bob.
2. Bob verifies Alice's signature, chooses a random r_B and a random session key k.

THREE-WAY AUTHENTICATION and also KEY-AGREEMENT I

A PKC will be used with encryption/decryption algorithms (e_U, d_U), for each user U, and DSS with signing/verification algorithms(s_U, v_U). Alice and Bob will have their, public, identity strings I_A and I_B.

1. Alice chooses a random integer r_A, sets $t = (I_B,\ r_A)$, signs it as $sig_{s_A}(t)$ and sends $m_1 = (t, sig_{s_A}(t))$ to Bob.
2. Bob verifies Alice's signature, chooses a random r_B and a random session key k.He then encrypts k with Alice's public key to get $E_{e_A}(k) = c$, sets

$$t_1 = (I_A, r_A, r_B, c),$$

and signs it as $sig_{s_B}(t_1)$. Then he sends $m_2 = (t_1, sig_{s_B}(t_1))$ to Alice.

THREE-WAY AUTHENTICATION and KEY AGREEMENT II

3. Alice verifies Bob's signature $sig_{s_B}(t_1)$ with $t_1 = (I_A, r_A, r_B, c)$,, and then checks that the r_A she just got matches the one she generated in Step 1.

3. Alice verifies Bob's signature $sig_{s_B}(t_1)$ with $t_1 = (I_A, r_A, r_B, c)$,, and then checks that the r_A she just got matches the one she generated in Step 1.
Once verified, she is convinced that she is communicating with Bob. She gets session key k via

$$D_{d_A}(c) = D_{d_A}(E_{e_A}(k)) = k,$$

sets $t_2 = (I_B, r_B)$ and signs it as $sig_{s_A}(t_2)$. Then she sends $m_3 = (t_2, sig_{s_A}(t_2))$ to Bob.

THREE-WAY AUTHENTICATION and KEY AGREEMENT II

3. Alice verifies Bob's signature $sig_{s_B}(t_1)$ with $t_1 = (I_A, r_A, r_B, c)$,, and then checks that the r_A she just got matches the one she generated in Step 1.
   Once verified, she is convinced that she is communicating with Bob. She gets session key k via
   $$D_{d_A}(c) = D_{d_A}(E_{e_A}(k)) = k,$$
   sets $t_2 = (I_B, r_B)$ and signs it as $sig_{s_A}(t_2)$. Then she sends $m_3 = (t_2, sig_{s_A}(t_2))$ to Bob.
4. Bob verifies Alice's signature and checks that r_B he just got matches his choice in Step 2. If both verifications pass, Alice and Bob have mutually authenticated each other's identity and, in addition, have agreed upon a session key k.

The goal of data authentication schemes (protocols) is to handle the case that data are sent through insecure channels.

The goal of data authentication schemes (protocols) is to handle the case that data are sent through insecure channels.

By creating so-called Message Authentication Code (MAC) a sending this MAC, together with a message through an insecure channel, one can create possibility to verify whether data were not changed in the channel.

The goal of data authentication schemes (protocols) is to handle the case that data are sent through insecure channels.

By creating so-called Message Authentication Code (MAC) a sending this MAC, together with a message through an insecure channel, one can create possibility to verify whether data were not changed in the channel.

The price to pay is that communicating parties need to share a secret random key that needs to be transmitted through a secure channel.

Basic difference between MACs and digital signatures is that MACs are symmetric in the following sense: Anyone who is able to verify MAC of a message is also able to generate the same MAC, and vice verse.

SCHEMES for DATA AUTHENTICATION

Basic difference between MACs and digital signatures is that MACs are symmetric in the following sense: Anyone who is able to verify MAC of a message is also able to generate the same MAC, and vice verse.

A scheme (M, T, K) for data authentication is given by:

- M is a set of possible messages (data)
- T is a set of possible MACs – (tags)
- K is a set of possible keys

SCHEMES for DATA AUTHENTICATION

Basic difference between MACs and digital signatures is that MACs are symmetric in the following sense: Anyone who is able to verify MAC of a message is also able to generate the same MAC, and vice verse.

A scheme (M, T, K) for data authentication is given by:

- M is a set of possible messages (data)
- T is a set of possible MACs – (tags)
- K is a set of possible keys

Moreover, it is required that

- to each k from K there is a single and easy to compute authentication mapping

$$auth_k : \{0,1\}^* \times M \rightarrow T$$

- and a single and easy to compute verification mapping

$$ver_k : M \times T \rightarrow \{true, false\}$$

such that the following two conditions should be satisfied:

SCHEMES for DATA AUTHENTICATION

Basic difference between MACs and digital signatures is that MACs are symmetric in the following sense: Anyone who is able to verify MAC of a message is also able to generate the same MAC, and vice verse.

A scheme (M, T, K) for data authentication is given by:

- M is a set of possible messages (data)
- T is a set of possible MACs – (tags)
- K is a set of possible keys

Moreover, it is required that

- to each k from K there is a single and easy to compute authentication mapping

$$auth_k : \{0,1\}^* \times M \rightarrow T$$

- and a single and easy to compute verification mapping

$$ver_k : M \times T \rightarrow \{true, false\}$$

such that the following two conditions should be satisfied:

Correctness: For each m from M and k from K it holds $ver_k(m, c) = true$, if there exists an r from $\{0, 1\}^*$ such that $c = auth_k(r, m)$

Security: For any $m \in M$ and any $k \in K$ it is computationally unfeasible, without a knowledge of k, to find $t \in T$ such that $ver_k(m, t) = true$

FROM BLOCK CIPHERS to MAC – CBC-MAC

Let C be an encryption algorithm that maps k-bit strings into k-bit strings.

FROM BLOCK CIPHERS to MAC – CBC-MAC

Let C be an encryption algorithm that maps k-bit strings into k-bit strings.

If a message

$$m = m_1 m_2 \dots m_l$$

is divided into blocks of length k, then so-called CBC-mode of encryption assumes a choice (random) of a special block y_0 of length k, and performs the following computations for i = 1, . . . ,l

$$y_i = C(y_{i-1} \oplus m_i)$$

and then

$$y_1 \| y_2 \| \dots \| y_l$$

is the encryption of m and

y_l can then be considered as the MAC for m.

A modification of this method is to use another crypto-algorithm to encrypt the last block m_l.

SPECIAL WEAKNESS of the CBS-MAC METHOD

Let us have three pairs and in each pair a message and its MAC

$$(m_1, t_1), (m_2, t_2), (m_3, t_3)$$

where messages m_1, m_3 and also t_1, t_3 are also of the length k and

$$m_2 = m_1 \| B \| m_2'$$

for some B that is also of length k. The encryption of the block B within m_2 is $C(B \oplus t_1)$.

SPECIAL WEAKNESS of the CBS-MAC METHOD

Let us have three pairs and in each pair a message and its MAC

$$(m_1, t_1), (m_2, t_2), (m_3, t_3)$$

where messages m_1, m_3 and also t_1, t_3 are also of the length k and

$$m_2 = m_1 \| B \| m_2'$$

for some B that is also of length k. The encryption of the block B within m_2 is $C(B \oplus t_1)$.

If we now define

$$B' = B \oplus t_1 \oplus t_3,\ m_4 = m_3 \| B' \| m_2'\ ,$$

then, during the encryption of m_4, we get

$$C(B' \oplus t_3) = C(B \oplus t_1),$$

SPECIAL WEAKNESS of the CBS-MAC METHOD

Let us have three pairs and in each pair a message and its MAC

$$(m_1, t_1), (m_2, t_2), (m_3, t_3)$$

where messages m_1, m_3 and also t_1, t_3 are also of the length k and

$$m_2 = m_1 \| B \| m_2'$$

for some B that is also of length k. The encryption of the block B within m_2 is $C(B \oplus t_1)$.

If we now define

$$B' = B \oplus t_1 \oplus t_3,\ m_4 = m_3 \| B' \| m_2'\ ,$$

then, during the encryption of m_4, we get

$$C(B' \oplus t_3) = C(B \oplus t_1),$$

This implies that MAC's for m_4 and m_2 are the same. One can therefore forge a new valid pair

$$(m_4, t_2).$$

ANALYSIS of CBC-MAC – a view

Theorem Given are two independent random permutations C_1 and C_2 on the set of message blocks M of cardinality n. Let us define

$$MAC(m_1, m_2, \ldots, m_l) = C_2(C_1(\ldots C_1(C_1(m_1) \oplus m_2) \oplus \ldots \oplus)m_{l-1}) \oplus m_l).$$

Let us assume that the MAC function is implemented by an oracle, and consider an adversary who can send queries to the oracle with a limited total length of q. Let $m_1, \ldots, m_d$ denote the finite block sequences on **M** which are sent by the adversary to the oracle and let the total number of blocks be less than q. Let the purpose of the adversary be to output a message m which is different from all m_i together with its MAC value c. Then the probability of success of the adversary (i.e. the probability that his MAC value is correct) is smaller than

$$\frac{q(q+1)}{2} \times \frac{1}{n-q} + \frac{1}{n-d}.$$

When $q = \theta n^{\frac{1}{2}}$, this is approximately a$= \frac{\theta^2}{2}$ (which is greater than $1 - e^{-a}$)

Implication: if the total length of all authenticated messages is negligible against # n, then there is no better way than the brute force attack to get collisions on the CBC-MAC.

FROM HASH FUNCTIONS TO HMAC

So called HMAC was published as the internet standard RFC2104.

Let a hash function h process messages by blocks of b bytes and produce a digest of l bytes and let t be the size of MAC, in bytes. HMAC of a message m with a key k is computed as follows:

- If k has more than b bytes replace k with h(k).
- Append zero bytes to k to have exactly b bytes.
- Compute (using constant strings opad and ipad)

$$h(k \oplus opad \| h(k \oplus ipad \| m)).$$

  and truncate the results to its t leftmost bytes to get $HMAC_k(m)$.

There is a variety of HMAC systems and they are usually specified by hash function that is used

SECURITY of HMAC

It can be shown that if

- $h(k \oplus ipad \| m)$ defines a secure MAC on fixed length messages, and
- h is collision free,

then HMAC is a secure MAC on variable length messages with two independent keys.
More precisely:

SECURITY of HMAC

It can be shown that if

- $h(k \oplus ipad \| m)$ defines a secure MAC on fixed length messages, and
- h is collision free,

then HMAC is a secure MAC on variable length messages with two independent keys. More precisely:

Theorem Let h be a hash function which hashes into l bits. Given k_1, k_2 from $\{0,1\}^l$ consider the following MAC algorithm

$$MAC_{k_1,k_2}(m) = h(k_2 \| h(k_1 \| m))$$

If h is collision free and $m \rightarrow h(k_2 \| m)$ is a secure MAC algorithm for messages m of the fixed length l, then the HMAC is a secure MAC algorithm for messages of arbitrary length.

DISADVANTAGE of STATIC USER IDENTIFICATION SCHEMES

Everybody who knows your password or PIN can impersonate you.

DISADVANTAGE of STATIC USER IDENTIFICATION SCHEMES

Everybody who knows your password or PIN can impersonate you.

Better are dynamic means of identification - for example challenge and response protocols.

DISADVANTAGE of STATIC USER IDENTIFICATION SCHEMES

Everybody who knows your password or PIN can impersonate you.

Better are dynamic means of identification - for example challenge and response protocols.
Basic idea.

- Alice claims ability to solve some hard problem P.

DISADVANTAGE of STATIC USER IDENTIFICATION SCHEMES

Everybody who knows your password or PIN can impersonate you.

Better are dynamic means of identification - for example challenge and response protocols.
Basic idea.

- Alice claims ability to solve some hard problem P.
- Bob challenges her ability by asking her to solve a particular instance of the P problem.

DISADVANTAGE of STATIC USER IDENTIFICATION SCHEMES

Everybody who knows your password or PIN can impersonate you.

Better are dynamic means of identification - for example challenge and response protocols.
Basic idea.

- Alice claims ability to solve some hard problem P.
- Bob challenges her ability by asking her to solve a particular instance of the P problem.
- If she succeeds, then Bob intends to believe he is indeed communicating with Alice.

DISADVANTAGE of STATIC USER IDENTIFICATION SCHEMES

Everybody who knows your password or PIN can impersonate you.

Better are dynamic means of identification - for example challenge and response protocols.
Basic idea.

- Alice claims ability to solve some hard problem P.
- Bob challenges her ability by asking her to solve a particular instance of the P problem.
- If she succeeds, then Bob intends to believe he is indeed communicating with Alice.

Using so called zero-knowledge identification schemes, discussed in the next chapter, you can identify yourself without giving to the identificator the ability to impersonate you.

SIMPLIFIED Fiat-Shamir IDENTIFICATION SCHEME

A trusted authority (TA) chooses: large random primes p,q, computes n = pq; and chooses a quadratic residue $v \in QR_n$, and s such that $s^2 = v \pmod{n}$.

SIMPLIFIED Fiat-Shamir IDENTIFICATION SCHEME

A trusted authority (TA) chooses: large random primes p,q, computes $n = pq$; and chooses a quadratic residue $v \in QR_n$, and s such that $s^2 = v \pmod{n}$.

public-key: v

private-key: s (that Alice knows, but not Bob)

SIMPLIFIED Fiat-Shamir IDENTIFICATION SCHEME

A trusted authority (TA) chooses: large random primes p,q, computes $n = pq$; and chooses a quadratic residue $v \in QR_n$, and s such that $s^2 = v \pmod n$.

public-key: v

private-key: s (that Alice knows, but not Bob)

Challenge-response Identification protocol

1. Alice chooses a random $r < n$, computes $x = r^2 \mod n$ and sends x to Bob.

SIMPLIFIED Fiat-Shamir IDENTIFICATION SCHEME

A trusted authority (TA) chooses: large random primes p,q, computes $n = pq$; and chooses a quadratic residue $v \in QR_n$, and s such that $s^2 = v \pmod{n}$.

public-key: v

private-key: s (that Alice knows, but not Bob)

Challenge-response Identification protocol

1. Alice chooses a random $r < n$, computes $x = r^2 \mod n$ and sends x to Bob.
2. Bob sends to Alice a random bit (a challenge) b.

SIMPLIFIED Fiat-Shamir IDENTIFICATION SCHEME

A trusted authority (TA) chooses: large random primes p,q, computes $n = pq$; and chooses a quadratic residue $v \in QR_n$, and s such that $s^2 = v \pmod{n}$.

public-key: v

private-key: s (that Alice knows, but not Bob)

Challenge-response Identification protocol

1. Alice chooses a random $r < n$, computes $x = r^2 \mod n$ and sends x to Bob.
2. Bob sends to Alice a random bit (a challenge) b.
3. Alice sends Bob (a response) $y = rs^b \mod n$

SIMPLIFIED Fiat-Shamir IDENTIFICATION SCHEME

A trusted authority (TA) chooses: large random primes p,q, computes n = pq; and chooses a quadratic residue $v \in QR_n$, and s such that $s^2 = v \pmod{n}$.

public-key: v

private-key: s (that Alice knows, but not Bob)

Challenge-response Identification protocol

1. Alice chooses a random $r < n$, computes $x = r^2 \mod n$ and sends x to Bob.
2. Bob sends to Alice a random bit (a challenge) b.
3. Alice sends Bob (a response) $y = rs^b \mod n$
4. Bob identifies the sender as Alice if and only if $y^2 = xv^b \mod n$, which is taken as a proof that the sender knows square roots of x and of v.

SIMPLIFIED Fiat-Shamir IDENTIFICATION SCHEME

A trusted authority (TA) chooses: large random primes p,q, computes $n = pq$; and chooses a quadratic residue $v \in QR_n$, and s such that $s^2 = v \pmod{n}$.

public-key: v

private-key: s (that Alice knows, but not Bob)

Challenge-response Identification protocol

1. Alice chooses a random $r < n$, computes $x = r^2 \mod n$ and sends x to Bob.
2. Bob sends to Alice a random bit (a challenge) b.
3. Alice sends Bob (a response) $y = rs^b \mod n$
4. Bob identifies the sender as Alice if and only if $y^2 = xv^b \mod n$, which is taken as a proof that the sender knows square roots of x and of v.

This protocol is a so-called single accreditation protocol

Alice proves her identity by convincing Bob that she knows the square root s of v (without revealing s to Bob) and the square root r of x.

If protocol is repeated t times, Alice has a chance 2^{-t} to fool Bob if she does not know s and r.

ANALYSIS of Fiat-Shamir IDENTIFICATION I

public-key: v
private-key: s (of Alice) such that $s^2 = v \pmod n$.

Protocol

1. Alice chooses a random $r < n$, computes $x = r^2 \mod n$ and sends x (her commitment) to Bob.

public-key: v
private-key: s (of Alice) such that $s^2 = v \pmod{n}$.

Protocol

1. Alice chooses a random $r < n$, computes $x = r^2 \mod n$ and sends x (her commitment) to Bob.
2. Bob sends to Alice a random bit b (a challenge).
3. Alice sends to Bob (a response) $y = rs^b$.
4. Bob verifies if and only if $y^2 = xv^b \mod n$, proving that Alice knows a square root of x.

Analysis

1. The first message is a commitment by Alice that she knows square root of x.
2. The second message is a challenge by Bob.
   - If Bob sends $b = 0$, then Alice has to open her commitment and reveal r.
   - If Bob sends $b = 1$, the Alice has to show her secret s in an "encrypted form".
3. The third message is Alice's response to the challenge of Bob.

Analysis

1. The first message is a commitment by Alice that she knows square root of x.
2. The second message is a challenge by Bob.
   - If Bob sends $b = 0$, then Alice has to open her commitment and reveal r.
   - If Bob sends $b = 1$, the Alice has to show her secret s in an "encrypted form".
3. The third message is Alice's response to the challenge of Bob.

Completeness If Alice knows s, and both Alice and Bob follow the protocol, then the response rs^b is the square root of xv^b.

It can be shown that Eve can cheat with probability of success $\frac{1}{2}$ as follows:

- Eve chooses random $r \in Z_n^*$, random $b_1 \in \{0, 1\}$ and sends $x = r^2 v^{-b_1}$, to Bob.
- Bob chooses $b \in \{0, 1\}$ at random and sends it to Eve.
- Eve sends r to Bob.

HOW CAN BAD EVE CHEAT?

Eve can send, to fool Bob, as her commitment, either r^2 for a random r or $r^2 v^{-1}$

In the first case Eve can respond correctly to the Bob's challenge b=0, by sending r; but cannot respond correctly to the challenge $b = 1$.

In the second case Eve can respond correctly to Bob's challenge $b = 1$, by sending r again; but cannot respond correctly to the challenge $b = 0$.

Eve has therefore a 50% chance to cheat.

Fiat-Shamir IDENTIFICATION SCHEME – PARALLEL VERSION

In the following parallel version of Fiat-Shamir identification scheme the probability of a false identification is decreased.

Fiat-Shamir IDENTIFICATION SCHEME – PARALLEL VERSION

In the following parallel version of Fiat-Shamir identification scheme the probability of a false identification is decreased.

Choose primes p, q and compute $n = pq$ and choose as security parameters integers k, t.

Choose quadratic residues $v_1, \ldots, v_k \in QR_n$.

Compute $s_1, \ldots, s_k$ such that $s_i = \sqrt{v_i} \mod n$

Fiat-Shamir IDENTIFICATION SCHEME – PARALLEL VERSION

In the following parallel version of Fiat-Shamir identification scheme the probability of a false identification is decreased.

Choose primes p, q and compute $n = pq$ and choose as security parameters integers k, t.

Choose quadratic residues $v_1, \ldots, v_k \in QR_n$.

Compute $s_1, \ldots, s_k$ such that $s_i = \sqrt{v_i} \mod n$

public-key: $v_1, \ldots, v_k$ secret-key: $s_1, \ldots, s_k$ of Alice

Fiat-Shamir IDENTIFICATION SCHEME – PARALLEL VERSION

In the following parallel version of Fiat-Shamir identification scheme the probability of a false identification is decreased.

Choose primes p, q and compute $n = pq$ and choose as security parameters integers k, t.

Choose quadratic residues $v_1, \ldots, v_k \in QR_n$.

Compute $s_1, \ldots, s_k$ such that $s_i = \sqrt{v_i} \mod n$

public-key: $v_1, \ldots, v_k$ secret-key: $s_1, \ldots, s_k$ of Alice PROTOCOL:

1. Alice chooses a random $r < n$, computes $a = r^2 \mod n$ and sends a to Bob.

Fiat-Shamir IDENTIFICATION SCHEME – PARALLEL VERSION

In the following parallel version of Fiat-Shamir identification scheme the probability of a false identification is decreased.

Choose primes p, q and compute $n = pq$ and choose as security parameters integers k, t.

Choose quadratic residues $v_1, \ldots, v_k \in QR_n$.

Compute $s_1, \ldots, s_k$ such that $s_i = \sqrt{v_i} \mod n$

public-key: $v_1, \ldots, v_k$ secret-key: $s_1, \ldots, s_k$ of Alice PROTOCOL:

1. Alice chooses a random $r < n$, computes $a = r^2 \mod n$ and sends a to Bob.
2. Bob sends Alice a random k-bit string $b_1 \ldots b_k$.

Fiat-Shamir IDENTIFICATION SCHEME – PARALLEL VERSION

In the following parallel version of Fiat-Shamir identification scheme the probability of a false identification is decreased.

Choose primes p, q and compute $n = pq$ and choose as security parameters integers k, t.

Choose quadratic residues $v_1, \ldots, v_k \in QR_n$.

Compute $s_1, \ldots, s_k$ such that $s_i = \sqrt{v_i} \mod n$

public-key: $v_1, \ldots, v_k$ secret-key: $s_1, \ldots, s_k$ of Alice PROTOCOL:

1. Alice chooses a random $r < n$, computes $a = r^2 \mod n$ and sends a to Bob.
2. Bob sends Alice a random k-bit string $b_1 \ldots b_k$.
3. Alice sends to Bob

$$y = r \prod_{i=1}^{k} s_i^{b_i} \mod n$$

Fiat-Shamir IDENTIFICATION SCHEME – PARALLEL VERSION

In the following parallel version of Fiat-Shamir identification scheme the probability of a false identification is decreased.

Choose primes p, q and compute $n = pq$ and choose as security parameters integers k, t.

Choose quadratic residues $v_1, \ldots, v_k \in QR_n$.

Compute $s_1, \ldots, s_k$ such that $s_i = \sqrt{v_i} \mod n$

public-key: $v_1, \ldots, v_k$ secret-key: $s_1, \ldots, s_k$ of Alice PROTOCOL:

1. Alice chooses a random $r < n$, computes $a = r^2 \mod n$ and sends a to Bob.
2. Bob sends Alice a random k-bit string $b_1 \ldots b_k$.
3. Alice sends to Bob
$$y = r \prod_{i=1}^{k} s_i^{b_i} \mod n$$
4. Bob accepts if and only if
$$y^2 = a \prod_{i=1}^{k} v_i^{b_i} \mod n$$

Fiat-Shamir IDENTIFICATION SCHEME – PARALLEL VERSION

In the following parallel version of Fiat-Shamir identification scheme the probability of a false identification is decreased.

Choose primes p, q and compute $n = pq$ and choose as security parameters integers k, t.

Choose quadratic residues $v_1, \ldots, v_k \in QR_n$.

Compute $s_1, \ldots, s_k$ such that $s_i = \sqrt{v_i} \mod n$

public-key: $v_1, \ldots, v_k$ secret-key: $s_1, \ldots, s_k$ of Alice PROTOCOL:

1. Alice chooses a random $r < n$, computes $a = r^2 \mod n$ and sends a to Bob.
2. Bob sends Alice a random k-bit string $b_1 \ldots b_k$.
3. Alice sends to Bob
$$y = r \prod_{i=1}^{k} s_i^{b_i} \mod n$$
4. Bob accepts if and only if
$$y^2 = a \prod_{i=1}^{k} v_i^{b_i} \mod n$$

Alice and Bob repeat this protocol t times, until Bob is convinced that Alice knows $s_1, \ldots, s_k$.

The chance that Alice can fool Bob is 2^{-kt}, a significant decrease comparing with the chance $\frac{1}{2}$ of the previous version of the identification scheme.

THE SCHNORR IDENTIFICATION SCHEME – SETTING

This is a practically attractive because being computationally efficient (in time, space + communication) scheme which minimizes storage + computations performed by Alice (to be, for example, a smart card).

THE SCHNORR IDENTIFICATION SCHEME – SETTING

This is a practically attractive because being computationally efficient (in time, space + communication) scheme which minimizes storage + computations performed by Alice (to be, for example, a smart card).

Scheme also requires a trusted authority (TA) who

1. chooses: a large prime $p < 2^{512}$,
   a large prime q dividing p - 1 and $q \leq 2^{140}$,
   an $\alpha \in Z_p^*$ of order q,
   a security parameter t such that $2^t < q$,
   p, q, α, t are made public.
2. establishes: a secure digital signature scheme with a secret signing algorithm sig_{TA} and a public verification algorithm ver_{TA}.

THE SCHNORR IDENTIFICATION SCHEME – SETTING

This is a practically attractive because being computationally efficient (in time, space + communication) scheme which minimizes storage + computations performed by Alice (to be, for example, a smart card).

Scheme also requires a trusted authority (TA) who

1. chooses: a large prime $p < 2^{512}$,
   a large prime q dividing p - 1 and $q \leq 2^{140}$,
   an $\alpha \in Z_p^*$ of order q,
   a security parameter t such that $2^t < q$,
   p, q, α, t are made public.
2. establishes: a secure digital signature scheme with a secret signing algorithm sig_{TA} and a public verification algorithm ver_{TA}.

Protocol for issuing a certificate to Alice

1. TA establishes Alice's identity by conventional means and forms a 512-bit string ID(Alice) which contains the identification information.
2. Alice chooses a secret random $0 \leq a \leq q - 1$ and computes
   $$v = \alpha^{-a} \mod p$$
   and sends v to the TA.
3. TA generates signature
   $$s = sig_{\mathrm{TA}}(ID(Alice), v)$$
   and sends to Alice as hercertificate: C (Alice) = (ID(Alice), v, s)

Schnorr IDENTIFICATION SCHEME - PROTOCOL

1. Alice chooses a random $0 \leq k < q$ and computes
$$\gamma = \alpha^k \mod p.$$

Schnorr IDENTIFICATION SCHEME - PROTOCOL

1. Alice chooses a random $0 \leq k < q$ and computes
$$\gamma = \alpha^k \mod p.$$
2. Alice sends to Bob her certificate C (Alice) = (ID(Alice), v, s) and also γ.

Schnorr IDENTIFICATION SCHEME - PROTOCOL

1. Alice chooses a random $0 \leq k < q$ and computes
$$\gamma = \alpha^k \mod p.$$
2. Alice sends to Bob her certificate $\mathrm{C\ (Alice) = (ID(Alice),\ v,\ s)}$ and also γ.
3. Bob verifies the signature of TA by checking that
$$ver_{\mathrm{TA}}(ID(Alice), v, s) = true.$$

Schnorr IDENTIFICATION SCHEME - PROTOCOL

1. Alice chooses a random $0 \leq k < q$ and computes
$$\gamma = \alpha^k \mod p.$$
2. Alice sends to Bob her certificate C (Alice) = (ID(Alice), v, s) and also γ.
3. Bob verifies the signature of TA by checking that
$$ver_{\mathrm{TA}}(ID(Alice), v, s) = true.$$
4. Bob chooses a random $1 \leq r \leq 2^t$, where $t < \lg q$ is a security parameter and sends it to Alice (often $t \leq 40$).

Schnorr IDENTIFICATION SCHEME - PROTOCOL

1. Alice chooses a random $0 \leq k < q$ and computes
$$\gamma = \alpha^k \mod p.$$
2. Alice sends to Bob her certificate $\mathrm{C}\ (\mathrm{Alice}) = (\mathrm{ID}(\mathrm{Alice}),\ \mathrm{v},\ \mathrm{s})$ and also γ.
3. Bob verifies the signature of TA by checking that
$$ver_{\mathrm{TA}}(ID(Alice), v, s) = true.$$
4. Bob chooses a random $1 \leq r \leq 2^t$, where $t < \lg q$ is a security parameter and sends it to Alice (often $t \leq 40$).
5. Alice computes and sends to Bob
$$y = (k + ar) \mod p.$$

Schnorr IDENTIFICATION SCHEME - PROTOCOL

1. Alice chooses a random $0 \leq k < q$ and computes
$$\gamma = \alpha^k \mod p.$$
2. Alice sends to Bob her certificate C (Alice) = (ID(Alice), v, s) and also γ.
3. Bob verifies the signature of TA by checking that
$$ver_{\mathrm{TA}}(ID(Alice), v, s) = true.$$
4. Bob chooses a random $1 \leq r \leq 2^t$, where $t < \lg q$ is a security parameter and sends it to Alice (often $t \leq 40$).
5. Alice computes and sends to Bob
$$y = (k + ar) \mod p.$$
6. Bob verifies that
$$\gamma \equiv \alpha^y v^r \mod q$$

Schnorr IDENTIFICATION SCHEME - PROTOCOL

1. Alice chooses a random $0 \leq k < q$ and computes
$$\gamma = \alpha^k \mod p.$$
2. Alice sends to Bob her certificate C (Alice) = (ID(Alice), v, s) and also γ.
3. Bob verifies the signature of TA by checking that
$$ver_{\mathrm{TA}}(ID(Alice), v, s) = true.$$
4. Bob chooses a random $1 \leq r \leq 2^t$, where $t < \lg q$ is a security parameter and sends it to Alice (often $t \leq 40$).
5. Alice computes and sends to Bob
$$y = (k + ar) \mod p.$$
6. Bob verifies that
$$\gamma \equiv \alpha^y v^r \mod q$$
7. This way Alice proofs her identity to Bob. Indeed,
$$\begin{aligned} \alpha^y v^r &\equiv \alpha^{k+ar}\alpha^{-ar} \mod p \\ &\equiv \alpha^k \mod p \\ &\equiv \gamma \mod p. \end{aligned}$$

Total storage needed: 512 bits for ID(Alice), 512 bits for v, 320 bits for s (if DSS is used). In total – 1344 bits.

Total communication needed from: Alice $\rightarrow$ Bob – 1996 (= 1344+512+140) bits,
Bob $\rightarrow$ Alice 40 bits (to send r).

Okamoto IDENTIFICATION SCHEME

The disadvantage of the Schnorr identification scheme is that there is no proof of its security. For the following modification of the Schnorr identification scheme presented below, for the Okamoto identification scheme, a proof of security exists.

Okamoto IDENTIFICATION SCHEME

The disadvantage of the Schnorr identification scheme is that there is no proof of its security. For the following modification of the Schnorr identification scheme presented below, for the Okamoto identification scheme, a proof of security exists.

Basic setting: To set up the scheme TA chooses:

- a large prime $p \leq 2^{512}$,
- a large prime $q \geq 2^{140}$ dividing p - 1;
- two elements $\alpha_1, \alpha_2 \in Z_p^*$ of the order q.

TA makes public p, q, α_1, α_2 and keeps secret (also before Alice and Bob)

$$c = lg_{\alpha_1} \alpha_2.$$

Finally, TA chooses a signature scheme and a hash function.

Okamoto IDENTIFICATION SCHEME

The disadvantage of the Schnorr identification scheme is that there is no proof of its security. For the following modification of the Schnorr identification scheme presented below, for the Okamoto identification scheme, a proof of security exists.

Basic setting: To set up the scheme TA chooses:

- a large prime $p \leq 2^{512}$,
- a large prime $q \geq 2^{140}$ dividing p - 1;
- two elements $\alpha_1, \alpha_2 \in Z_p^*$ of the order q.

TA makes public p, q, α_1, α_2 and keeps secret (also before Alice and Bob)

$$c = lg_{\alpha_1} \alpha_2.$$

Finally, TA chooses a signature scheme and a hash function.

Issuing a certificate to Alice

- TA establishes Alice's identity and issues her identification string ID(Alice).
- Alice secretly and randomly chooses $0 \leq a_1, a_2 \leq q-1$ and sends to TA
$$v = \alpha_1^{-a_1} \alpha_2^{-a_2} \mod p.$$
- TA generates a signature $s = sig_{\mathrm{TA}}(ID(Alice), v)$ and sends to Alice the certificate
C (Alice) = (ID(Alice), v, s).

Okamoto IDENTIFICATION SCHEME – BASICS ONCE MORE

Basic setting

TA chooses: a large prime $p \leq 2^{512}$,large prime $q \geq 2^{140}$ dividing p - 1; two elements $\alpha_1, \alpha_2 \in Z_p^*$ of order q. TA keep secret (also from Alice and Bob) $c = \lg_{\alpha_1} \alpha_2$.

Issuing a certificate to Alice

- TA establishes Alice's identity and issues an identification string ID(Alice).
- Alice randomly chooses $0 \leq a_1, a_2 \leq q - 1$ and sends to TA.
$$v = \alpha_1^{-a_1} \alpha_2^{-a_2} \mod p.$$
- TA generates a signature $s = sig_{\mathrm{TA}}(ID(Alice), v)$ and sends to Alice the certificate
C (Alice) = (ID(Alice), v, s).

Okamoto IDENTIFICATION SCHEME

- Alice chooses random $0 \leq k_1, k_2 \leq q-1$ and computes
$$\gamma = \alpha_1^{k_1}\alpha_2^{k_2} \mod p.$$

Okamoto IDENTIFICATION SCHEME

- Alice chooses random $0 \leq k_1, k_2 \leq q-1$ and computes
$$\gamma = \alpha_1^{k_1}\alpha_2^{k_2} \mod p.$$
- Alice sends to Bob her certificate (ID(Alice), v, s) and γ.

Okamoto IDENTIFICATION SCHEME

- Alice chooses random $0 \leq k_1, k_2 \leq q - 1$ and computes
$$\gamma = \alpha_1^{k_1} \alpha_2^{k_2} \mod p.$$
- Alice sends to Bob her certificate (ID(Alice), v, s) and γ.
- Bob verifies the signature of TA by checking that
$$ver_{TA}(ID(Alice), v, s) = true.$$

Okamoto IDENTIFICATION SCHEME

- Alice chooses random $0 \leq k_1, k_2 \leq q - 1$ and computes
$$\gamma = \alpha_1^{k_1} \alpha_2^{k_2} \mod p.$$
- Alice sends to Bob her certificate (ID(Alice), v, s) and γ.
- Bob verifies the signature of TA by checking that
$$ver_{TA}(ID(Alice), v, s) = true.$$
- Bob chooses a random $1 \leq r \leq 2^t$ and sends it to Alice.

Okamoto IDENTIFICATION SCHEME

- Alice chooses random $0 \leq k_1, k_2 \leq q-1$ and computes
$$\gamma = \alpha_1^{k_1}\alpha_2^{k_2} \mod p.$$
- Alice sends to Bob her certificate (ID(Alice), v, s) and γ.
- Bob verifies the signature of TA by checking that
$$ver_{TA}(ID(Alice), v, s) = true.$$
- Bob chooses a random $1 \leq r \leq 2^t$ and sends it to Alice.
- Alice sends to Bob
$$y_1 = (k_1 + a_1 r) \mod q;\ y_2 = (k_2 + a_2 r) \mod q.$$

Okamoto IDENTIFICATION SCHEME

- Alice chooses random $0 \leq k_1, k_2 \leq q-1$ and computes
$$\gamma = \alpha_1^{k_1}\alpha_2^{k_2} \mod p.$$
- Alice sends to Bob her certificate (ID(Alice), v, s) and γ.
- Bob verifies the signature of TA by checking that
$$ver_{TA}(ID(Alice), v, s) = true.$$
- Bob chooses a random $1 \leq r \leq 2^t$ and sends it to Alice.
- Alice sends to Bob
$$y_1 = (k_1 + a_1 r) \mod q;\ y_2 = (k_2 + a_2 r) \mod q.$$
- Bob verifies
$$\gamma \equiv \alpha_1^{y_1}\alpha_2^{y_2} v^r \pmod p$$

AUTHENTICATION CODES

They provide methods to ensure integrity of messages – that a message has not been tampered/changed, and that the message originated with the presumed sender.

The goal is to achieve authentication even in the presence of Mallot, a man in the middle, who can observe transmitted messages and replace them by messages of his own choice.

Formally, an authentication code consists of:

- A set M of possible messages.
- A set T of possible authentication tags.
- A set K of possible keys.
- A set R of authentication algorithms $a_k : M \to T$, one for each $k \in K$

AUTHENTICATION CODES

They provide methods to ensure integrity of messages – that a message has not been tampered/changed, and that the message originated with the presumed sender.
The goal is to achieve authentication even in the presence of Mallot, a man in the middle, who can observe transmitted messages and replace them by messages of his own choice.

Formally, an authentication code consists of:

- A set M of possible messages.
- A set T of possible authentication tags.
- A set K of possible keys.
- A set R of authentication algorithms $a_k : M \rightarrow T$, one for each $k \in K$

Transmission process

- Alice and Bob jointly choose a secret key k.
- If Alice wants to send a message w to Bob, she sends (w, t), where $t = a_k(w)$.
- If Bob receives (w, t) he computes $t' = a_k(w)$ and if t = t', then Bob accepts the message w as authentic.

ATTACKS and DECEPTION PROBABILITIES

There are two basic types of attacks Mallot, the man in the middle, can do.

ATTACKS and DECEPTION PROBABILITIES

There are two basic types of attacks Mallot, the man in the middle, can do.

Impersonation. Mallot introduces a message (w, t) into the channel – expecting that message will be received as being sent by Alice.

ATTACKS and DECEPTION PROBABILITIES

There are two basic types of attacks Mallot, the man in the middle, can do.

Impersonation. Mallot introduces a message (w, t) into the channel – expecting that message will be received as being sent by Alice.

Substitution. Mallot replaces a message (w, t) in the channel by another one, (w', t') – expecting that message will be accepted as being sent by Alice.

ATTACKS and DECEPTION PROBABILITIES

There are two basic types of attacks Mallot, the man in the middle, can do.

Impersonation. Mallot introduces a message (w, t) into the channel – expecting that message will be received as being sent by Alice.

Substitution. Mallot replaces a message (w, t) in the channel by another one, (w', t') – expecting that message will be accepted as being sent by Alice.

With any impersonation (substitution) attack a probability $P_i(P_s)$ is associated that Mallot will deceive Bob, if Mallot follows an optimal strategy.

ATTACKS and DECEPTION PROBABILITIES

There are two basic types of attacks Mallot, the man in the middle, can do.

Impersonation. Mallot introduces a message (w, t) into the channel – expecting that message will be received as being sent by Alice.

Substitution. Mallot replaces a message (w, t) in the channel by another one, (w', t') – expecting that message will be accepted as being sent by Alice.

With any impersonation (substitution) attack a probability $P_i(P_s)$ is associated that Mallot will deceive Bob, if Mallot follows an optimal strategy.
In order to determine such probabilities we need to know probability distributions p_m on messages and p_k on keys.

ATTACKS and DECEPTION PROBABILITIES

There are two basic types of attacks Mallot, the man in the middle, can do.

Impersonation. Mallot introduces a message (w, t) into the channel – expecting that message will be received as being sent by Alice.

Substitution. Mallot replaces a message (w, t) in the channel by another one, (w', t') – expecting that message will be accepted as being sent by Alice.

With any impersonation (substitution) attack a probability $P_i(P_s)$ is associated that Mallot will deceive Bob, if Mallot follows an optimal strategy.

In order to determine such probabilities we need to know probability distributions p_m on messages and p_k on keys.

In the following so called authentication matrices $|K| \times |M|$ will tabulate all authentication tags. The item in a row corresponding to a key k and in a column corresponding to a message w will contain the authentication tag $t_k(w)$.

ATTACKS and DECEPTION PROBABILITIES

There are two basic types of attacks Mallot, the man in the middle, can do.

Impersonation. Mallot introduces a message (w, t) into the channel – expecting that message will be received as being sent by Alice.

Substitution. Mallot replaces a message (w, t) in the channel by another one, (w', t') – expecting that message will be accepted as being sent by Alice.

With any impersonation (substitution) attack a probability $P_i(P_s)$ is associated that Mallot will deceive Bob, if Mallot follows an optimal strategy.

In order to determine such probabilities we need to know probability distributions p_m on messages and p_k on keys.

In the following so called authentication matrices $|K| \times |M|$ will tabulate all authentication tags. The item in a row corresponding to a key k and in a column corresponding to a message w will contain the authentication tag $t_k(w)$.

The goal of authentication codes, to be discussed next, is to decrease probabilities that Mallot performs successfully impersonation or substitution.

EXAMPLE

Let $M = T = Z_3$, $K = Z_3 \times Z_3$.
For $(i,j) \in K$ and $w \in M$, let $t_{ij}(w) = (iw + j) \mod 3$.

EXAMPLE

Let $M = T = Z_3$, $K = Z_3 \times Z_3$.
For $(i, j) \in K$ and $w \in M$, let $t_{ij}(w) = (iw + j) \mod 3$.
Let the matrix key $\times$ message of authentication tags has the form

Key	0	1	2
(0,0)	0	0	0
(0,1)	1	1	1
(0,2)	2	2	2
(1,0)	0	1	2
(1,1)	1	2	0
(1,2)	2	0	1
(2,0)	0	2	1
(2,1)	1	0	2
(2,2)	2	1	0

Impersonation attack: Mallot picks a message w and tries to guess the correct authentication tag.
However, for each message w and each tag a there are exactly three keys k such that $t_k(w) = a$. Hence $P_i = \frac{1}{3}$.

EXAMPLE

Let $M = T = Z_3$, $K = Z_3 \times Z_3$.
For $(i, j) \in K$ and $w \in M$, let $t_{ij}(w) = (iw + j) \mod 3$.
Let the matrix key $\times$ message of authentication tags has the form

Key	0	1	2
(0,0)	0	0	0
(0,1)	1	1	1
(0,2)	2	2	2
(1,0)	0	1	2
(1,1)	1	2	0
(1,2)	2	0	1
(2,0)	0	2	1
(2,1)	1	0	2
(2,2)	2	1	0

Impersonation attack: Mallot picks a message w and tries to guess the correct authentication tag.
However, for each message w and each tag a there are exactly three keys k such that $t_k(w) = a$. Hence $P_i = \frac{1}{3}$.
Substitution attack: By checking the table one can see that if Mallot observes an authenticated message (w, t), then there are only three possibilities for the key that was used.
Moreover, for each choice (w', t'), w $\neq$ w', there is exactly one of the three possible keys for (w',t') that can be used. Therefore $P_s = \frac{1}{3}$.

ORTHOGONAL ARRAYS

Definition An orthogonal array OA(n, k, λ) is a $\lambda n^2 \times k$ array of n symbols, such that in any two columns of the array every one of the possible n^2 pairs of symbols occurs in exactly λ rows.

Example OA(3,3,1) obtained from the authentication matrix presented before;

$$\begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 2 & 2 & 2 \\ 0 & 1 & 2 \\ 1 & 2 & 0 \\ 2 & 0 & 1 \\ 0 & 2 & 1 \\ 1 & 0 & 2 \\ 2 & 1 & 0 \end{pmatrix}$$

ORTHOGONAL ARRAYS

Definition An orthogonal array OA(n, k, λ) is a $\lambda n^2 \times k$ array of n symbols, such that in any two columns of the array every one of the possible n^2 pairs of symbols occurs in exactly λ rows.

Example OA(3,3,1) obtained from the authentication matrix presented before;

$$\begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 2 & 2 & 2 \\ 0 & 1 & 2 \\ 1 & 2 & 0 \\ 2 & 0 & 1 \\ 0 & 2 & 1 \\ 1 & 0 & 2 \\ 2 & 1 & 0 \end{pmatrix}$$

Theorem Suppose we have an orthogonal array OA(n, k, λ).Then there is an authentication code with $|M| = k, |A| = n, |K| = \lambda n^2$ and $P_I = P_s = \frac{1}{n}$.

Proof Use each row of the orthogonal array as an authentication rule (key) with equal probability. Therefore we have the following correspondence:

orthogonal array	authentication code
row	authentication rule
column	message
symbol	authentication tag

CONSTRUCTION and BOUNDS for OAs

In an orthogonal array OA(n, k, λ)

- n determines the number of authenticators (security of the code);
- k is the number of messages the code can accommodate;
- λ relates to the number of keys $-\lambda n^2$.

CONSTRUCTION and BOUNDS for OAs

In an orthogonal array OA(n, k, λ)

- n determines the number of authenticators (security of the code);
- k is the number of messages the code can accommodate;
- λ relates to the number of keys $-\lambda n^2$.

The following holds for orthogonal arrays.

- If p is prime, then OA(p, p, 1) exits.

CONSTRUCTION and BOUNDS for OAs

In an orthogonal array OA(n, k, λ)

- n determines the number of authenticators (security of the code);
- k is the number of messages the code can accommodate;
- λ relates to the number of keys $-\lambda n^2$.

The following holds for orthogonal arrays.

- If p is prime, then OA(p, p, 1) exits.
- Suppose there exists an OA(n, k, λ). Then

$$\lambda \geq \frac{k(n-1)+1}{n^2};$$

CONSTRUCTION and BOUNDS for OAs

In an orthogonal array OA(n, k, λ)

- n determines the number of authenticators (security of the code);
- k is the number of messages the code can accommodate;
- λ relates to the number of keys $-\lambda n^2$.

The following holds for orthogonal arrays.

- If p is prime, then OA(p, p, 1) exits.
- Suppose there exists an OA(n, k, λ). Then
$$\lambda \geq \frac{k(n-1)+1}{n^2};$$
- Suppose that p is a prime and $d \leq 2$ an integer. Then there is an orthogonal array $OA(p, \frac{(p^d-1)}{(p-1)}, p^{d-2})$.
- Let us have an authentication code with $|A| = n$ and $P_i = P_s = \frac{1}{n}$.Then $|K| \geq n^2$. Moreover, $|K| = n^2$ if and only if there is an orthogonal array OA(n, k,1), where $|M| = k$ and $P_K(k) = \frac{1}{n^2}$ for every key $k \in K$.

CONSTRUCTION and BOUNDS for OAs

In an orthogonal array OA(n, k, λ)

- n determines the number of authenticators (security of the code);
- k is the number of messages the code can accommodate;
- λ relates to the number of keys $-\lambda n^2$.

The following holds for orthogonal arrays.

- If p is prime, then OA(p, p, 1) exits.
- Suppose there exists an OA(n, k, λ). Then
$$\lambda \geq \frac{k(n-1)+1}{n^2};$$
- Suppose that p is a prime and $d \leq 2$ an integer. Then there is an orthogonal array $OA(p, \frac{(p^d-1)}{(p-1)}, p^{d-2})$.
- Let us have an authentication code with $|A| = n$ and $P_i = P_s = \frac{1}{n}$.Then $|K| \geq n^2$. Moreover, $|K| = n^2$ if and only if there is an orthogonal array OA(n, k,1), where $|M| = k$ and $P_K(k) = \frac{1}{n^2}$ for every key $k \in K$.

The last claim shows that there are no much better approaches to authentication codes with deception probabilities as small as possible than orthogonal arrays.

SECRET SHARING - PROBLEM

In many applications it is of importance to distribute a sensitive information, called here as a secret (for example an algorithm how to open a safe or a secret key) among several parties in such a way that only a well define subset of parties can determine the secret if they cooperate.

SECRET SHARING - PROBLEM

In many applications it is of importance to distribute a sensitive information, called here as a secret (for example an algorithm how to open a safe or a secret key) among several parties in such a way that only a well define subset of parties can determine the secret if they cooperate.

In some other cases one can increase security of confidential information, say a secret key, by sharing it between several parties.

SECRET SHARING - PROBLEM

In many applications it is of importance to distribute a sensitive information, called here as a secret (for example an algorithm how to open a safe or a secret key) among several parties in such a way that only a well define subset of parties can determine the secret if they cooperate.

In some other cases one can increase security of confidential information, say a secret key, by sharing it between several parties.

In the following we show how to solve this problem in the following "threshold" setting:

How to "partition" a number S (called here as a "secret") into n "shares" and distribute them among n parties in such a way that for a fixed (threshold) $t < n$ any t of them can create S, but no $t-1$, or less, of them can can the slightest idea how to do that.

BASIC IDEA of the (n,t) THRESHOLD SECRET SHARING

To distribute a secret (number) S among n parties, the dealer creates a degree $t-1$ random polynomial p such that p(0)=S and distributes to each party a "share" of it – value of p in a separate point.

Since each degree $t-1$ polynomial p is uniquely determined by any t points on p, the above distribution of points allows any t users to determine p, and so also p(0)=S, and no smaller group of parties, can have slightest idea about S.

A dealer creates shares of a binary-string secret s and distributes them between two parties P_1 and P_2 by choosing a random binary string b, of the same length as s, and

- sends the share b to P_1 and
- sends the share $s \oplus b$ to P_2.

SECRET SHARING between TWO PARTIES

A dealer creates shares of a binary-string secret s and distributes them between two parties P_1 and P_2 by choosing a random binary string b, of the same length as s, and

- sends the share b to P_1 and
- sends the share $s \oplus b$ to P_2.

This way, none of the parties P_1 and P_2 alone has a slightest idea about s, but both together easily recover s by computing

$$b \oplus (s \oplus b) = s.$$

The above scheme can be easily extended to the case of n users so that only all of them can reveal the secret.

THRESHOLD SECRET SHARING SCHEMES

Secret sharing schemes "partition" a "secret" into shares and distributes them among several parties in such a way that only predefined sets of parties can "assemble" the secret.

THRESHOLD SECRET SHARING SCHEMES

Secret sharing schemes "partition" a "secret" into shares and distributes them among several parties in such a way that only predefined sets of parties can "assemble" the secret.

For example, a vault in the bank can be opened only if at least two out of three responsible employees use their knowledge and tools (keys) to open the vault.

THRESHOLD SECRET SHARING SCHEMES

Secret sharing schemes "partition" a "secret" into shares and distributes them among several parties in such a way that only predefined sets of parties can "assemble" the secret.

For example, a vault in the bank can be opened only if at least two out of three responsible employees use their knowledge and tools (keys) to open the vault.

An important special simple case of secret sharing schemes are threshold secret sharing schemes at which a certain threshold of participant is needed and sufficient to assemble the secret.

THRESHOLD SECRET SHARING SCHEMES

Secret sharing schemes "partition" a "secret" into shares and distributes them among several parties in such a way that only predefined sets of parties can "assemble" the secret.

For example, a vault in the bank can be opened only if at least two out of three responsible employees use their knowledge and tools (keys) to open the vault.

An important special simple case of secret sharing schemes are threshold secret sharing schemes at which a certain threshold of participant is needed and sufficient to assemble the secret.

Definition Let $t \leq n$ be positive integers. A (n, t)-threshold scheme is a method of sharing a secret S among a set P of n parties, $P = \{P_i \,|\, 1 \leq i \leq n\}$, in such a way that any t, or more, parties can compute the value S , but no group of t - 1, or less, parties can compute S .

Secret S is chosen by a "dealer" $D \notin P$.

It is assumed that the dealer "distributes" the secret through shares to parties secretly and in such a way that no party knows shares of other parties.

Shamir's (n,t)-THRESHOLD SCHEME

Initial phase:

Dealer D chooses a prime p, n randomly chooses integers x_i, $1 \leq i \leq n$ and sends x_i to the user P_i.

The values x_i are then made public.

Shamir's (n,t)-THRESHOLD SCHEME

Initial phase:
Dealer D chooses a prime p, n randomly chooses integers x_i, $1 \leq i \leq n$ and sends x_i to the user P_i.
The values x_i are then made public.
Share distribution: Suppose that the dealer D wants to distribute a secret $S \in Z_p$ among n parties. D randomly chooses, and keeps secret, t - 1 elements of Z_p, $a_1, \ldots, a_{t-1}$.
For $1 \leq i \leq n$, D computes the "shares" $y_i = a(x_i)$,
where

$$a(x) = S + \sum_{j=1}^{t-1} a_j x^j \mod p.$$

D then sends the share y_i to the party P_i, $1 \leq i \leq n$ and keeps coefficients a_i secret.

Shamir's (n,t)-THRESHOLD SCHEME

Initial phase:
Dealer D chooses a prime p, n randomly chooses integers x_i, $1 \leq i \leq n$ and sends x_i to the user P_i.
The values x_i are then made public.
Share distribution: Suppose that the dealer D wants to distribute a secret $S \in Z_p$ among n parties. D randomly chooses, and keeps secret, t - 1 elements of Z_p, $a_1, \ldots, a_{t-1}$.
For $1 \leq i \leq n$, D computes the "shares" $y_i = a(x_i)$,
where

$$a(x) = S + \sum_{j=1}^{t-1} a_j x^j \mod p.$$

D then sends the share y_i to the party P_i, $1 \leq i \leq n$ and keeps coefficients a_i secret.
Secret accumulation: Let parties $P_{i_1}, \ldots, P_{i_t}$ want to determine secret S. Since, unknown to them, polynomial a(x) has degree t-1, a(x) they know that it has the form

$$a(x) = a_0 + a_1 x + \ldots + a_{t-1} x^{t-1},$$

and therefore they can determine all coefficients a_i from t equations $a(x_{i_j}) = y_{i_j}$, where all arithmetic is done modulo p.

It can be easily shown that equations obtained this way are linearly independent and the system has a unique solution.

Shamir's (n,t)-THRESHOLD SCHEME

Initial phase:
Dealer D chooses a prime p, n randomly chooses integers x_i, $1 \leq i \leq n$ and sends x_i to the user P_i.
The values x_i are then made public.
Share distribution: Suppose that the dealer D wants to distribute a secret $S \in Z_p$ among n parties. D randomly chooses, and keeps secret, t - 1 elements of Z_p, $a_1, \ldots, a_{t-1}$.
For $1 \leq i \leq n$, D computes the "shares" $y_i = a(x_i)$,
where

$$a(x) = S + \sum_{j=1}^{t-1} a_j x^j \mod p.$$

D then sends the share y_i to the party P_i, $1 \leq i \leq n$ and keeps coefficients a_i secret.
Secret accumulation: Let parties $P_{i_1}, \ldots, P_{i_t}$ want to determine secret S. Since, unknown to them, polynomial a(x) has degree t-1, a(x) they know that it has the form

$$a(x) = a_0 + a_1 x + \ldots + a_{t-1} x^{t-1},$$

and therefore they can determine all coefficients a_i from t equations $a(x_{i_j}) = y_{i_j}$, where all arithmetic is done modulo p.

It can be easily shown that equations obtained this way are linearly independent and the system has a unique solution.

In such a case $S = a_0$.

Shamir's SCHEME — TECHNICALITIES

Shamir's scheme uses the following result concerning polynomials over fields Z_p, where p is prime.

Theorem Let $f(x) = \sum_{i=0}^{t-1} a_i x^i \in Z_p[x]$ be a polynomial of degree t - 1 and let

$$\Omega = \{(x_i, f(x_i)) \,|\, x_i \in Z_p, i = 1, \ldots, t, x_i \neq x_j\}$$

if $i \neq j$. For any $Q \subseteq \Omega$, let $P_Q = \{g \in Z_p[x] | deg(g) = t-1, g(x) = y$ for all (x,y) $\in Q\}$. Then it holds:

- $P_S = \{f(x)\}$, i.e. f is the only polynomial of degree t - 1, whose graph contains all t points in Ω.
- If Q is a proper subset of Ω and $x \neq 0$ for all $(x, y) \in Q$, then each $a \in Z_p$ appears with the same frequency as the constant coefficient of polynomials in P_Q.

Shamir's SCHEME — TECHNICALITIES

Shamir's scheme uses the following result concerning polynomials over fields Z_p, where p is prime.

Theorem Let $f(x) = \sum_{i=0}^{t-1} a_i x^i \in Z_p[x]$ be a polynomial of degree t - 1 and let

$$\Omega = \{(x_i, f(x_i)) \,|\, x_i \in Z_p, i = 1, \ldots, t, x_i \neq x_j\}$$

if $i \neq j$. For any $Q \subseteq \Omega$, let $P_Q = \{g \in Z_p[x] | deg(g) = t-1, g(x) = y$ for all (x,y) $\in Q\}$. Then it holds:

- $P_S = \{f(x)\}$, i.e. f is the only polynomial of degree t - 1, whose graph contains all t points in Ω.
- If Q is a proper subset of Ω and $x \neq 0$ for all $(x, y) \in Q$, then each $a \in Z_p$ appears with the same frequency as the constant coefficient of polynomials in P_Q.

Corollary (Lagrange formula) Let $f(x) = \sum_{i=0}^{t-1} a_i x^i \in Z_p[x]$ be a polynomial and let $P = \{(x_I, f(x_i)) \,|\, i = 1, \ldots, t, x_i \neq x_j, i \neq j\}$. Then

$$f(x) = \sum_{i=1}^{t} f(x_i) \prod_{1 \leq j \leq t, \quad j \neq i} \frac{x - x_j}{x_i - x_j}$$

Shamir's (n,t)-THRESHOLD SCHEME — SUMMARY

To distribute n shares of a secret S among parties $P_1, \dots, P_n$ a dealer - a trusted authority TA proceeds as follows:

- TA chooses a prime $p > max\{S, n\}$ and sets $a_0 = S$.
- TA selects randomly $a_1, \dots, a_{t-1} \in Z_p$ and creates the polynomial $f(x) = \sum_{i=0}^{t-1} a_i x^i$.
- TA computes $s_i = f(i), i = 1, \dots, n$ and transfers each (i, s_i) to the party P_i in a secure way.

Any group J of t or more parties can compute the secret. Indeed, from the previous corollary we have

$$S = a_0 = f(0) = \sum_{i \in J} f(i) \prod_{j \in J, j \neq i} \frac{j}{j - i}$$

In case $|J| < t$, then each $a_0 \in Z_p$ is equally likely to be the secret.

SECRET SHARING – GENERAL CASE

A serious limitation of the threshold secret sharing schemes is that all groups of parties with the same number of parties have the same access to the secret.

SECRET SHARING – GENERAL CASE

A serious limitation of the threshold secret sharing schemes is that all groups of parties with the same number of parties have the same access to the secret.

Practical situations usually require that some (sets of) parties are more important than others.

SECRET SHARING – GENERAL CASE

A serious limitation of the threshold secret sharing schemes is that all groups of parties with the same number of parties have the same access to the secret.

Practical situations usually require that some (sets of) parties are more important than others.

Let P be a set of parties. To deal with the above situation such concepts as **authorized set of user** of P and **access structures** are used.

An authorized set of parties $A \subseteq P$ is a set of parties who can together construct the secret.

An unauthorized set of parties $U \subseteq P$ is a set of parties who alone cannot learn anything about the secret.

SECRET SHARING – GENERAL CASE

A serious limitation of the threshold secret sharing schemes is that all groups of parties with the same number of parties have the same access to the secret.

Practical situations usually require that some (sets of) parties are more important than others.

Let P be a set of parties. To deal with the above situation such concepts as **authorized set of user** of P and **access structures** are used.

An authorized set of parties $A \subseteq P$ is a set of parties who can together construct the secret.

An unauthorized set of parties $U \subseteq P$ is a set of parties who alone cannot learn anything about the secret.

Let P be a set of parties. The access structure $\Gamma \subseteq 2^P$ is a set such that $A \in \Gamma$ for all authorized sets A and $U \in 2^P - \Gamma$ for all unauthorized sets U.

Theorem: For any access structure there exists a secret sharing scheme realizing this access structure.

SECRET SHARING SCHEME with VERIFICATION

- Secret sharing protocols increase security of a secret information by sharing it between several parties.
- Some secret sharing scheme are such that they work even in case some parties behave incorrectly.

SECRET SHARING SCHEME with VERIFICATION

- Secret sharing protocols increase security of a secret information by sharing it between several parties.
- Some secret sharing scheme are such that they work even in case some parties behave incorrectly.
- A secret sharing scheme with verification is such a secret sharing scheme that:
  - Each P_i is capable to verify correctness of his/her share s_i
  - No party P_i is able to provide incorrect information and to convince others about its correctness

Feldman's (n,k)-PROTOCOL

Feldman's protocol is an example of the secret sharing scheme with verification. The protocol is a generalization of Shamir's protocol.

Feldman's (n,k)-PROTOCOL

Feldman's protocol is an example of the secret sharing scheme with verification. The protocol is a generalization of Shamir's protocol. It is assumed that all n participants can broadcast messages to all others and each of them can determine all senders.

Given are large primes $p, q, q|(p-1), q > n$ and $h < p$ – a generator of Z_p^* . All these numbers, and also the number $g = h^{\frac{p-1}{q}} \mod p$, are public.

Feldman's (n,k)-PROTOCOL

Feldman's protocol is an example of the secret sharing scheme with verification. The protocol is a generalization of Shamir's protocol. It is assumed that all n participants can broadcast messages to all others and each of them can determine all senders.

Given are large primes $p, q, q|(p-1), q > n$ and $h < p$ – a generator of Z_p^*. All these numbers, and also the number $g = h^{\frac{p-1}{q}} \mod p$, are public.

As in Shamir's scheme, to share a secret S, the dealer assigns to each party P_i a specific random x_i from $\{1, \ldots, q-1\}$ and generates a random secret polynomial

$$f(x) = \sum_{j=0}^{k-1} a_j x^j \mod q \qquad (1)$$

such that f(0) = S and sends to each P_i a value $y_i = f(x_i)$. In addition, using a broadcasting scheme, the dealer sends to each P_i all values $v_j = g^{a_j} \mod p$.

Each P_i verifies that

$$g^{y_i} = \prod_{j=0}^{k-1} (v_j)^{x_i^j} \mod p \qquad (1)$$

If (1) does not hold, P_i asks, using the broadcasting scheme, the dealer to broadcast correct value of y_i. If there are at least k such requests, or some of the new values of y_i does not satisfy (1), the dealer is considered as not reliable.

One can easily verify that if the dealer works correctly, then all relations (1) hold.

VISUAL SECRET SHARING

The basic idea is to create, for a visual information (a secret) S, a set of n transparencies in such a way that one can see S only if all n trancparencies are overlaid.

E-COMMERCE

Very important is to ensure security of e-money transactions needed for e-commerce.

Very important is to ensure security of e-money transactions needed for e-commerce.

In addition to providing security and privacy, the task is also to prevent alterations of purchase orders and forgery of credit card information.

Authenticity: Participants in transactions cannot be impersonated and signatures cannot be forged.

BASIC REQUIREMENTS for e-COMMERCE SYSTEMS

Authenticity: Participants in transactions cannot be impersonated and signatures cannot be forged.

Integrity: Documents (purchase orders, payment instructions,...) cannot be forged.

BASIC REQUIREMENTS for e-COMMERCE SYSTEMS

Authenticity: Participants in transactions cannot be impersonated and signatures cannot be forged.

Integrity: Documents (purchase orders, payment instructions,...) cannot be forged.

Privacy: Details of transaction should be kept secret.

BASIC REQUIREMENTS for e-COMMERCE SYSTEMS

Authenticity: Participants in transactions cannot be impersonated and signatures cannot be forged.

Integrity: Documents (purchase orders, payment instructions,...) cannot be forged.

Privacy: Details of transaction should be kept secret.

Security: Sensitive information (as credit card numbers) must be protected.

BASIC REQUIREMENTS for e-COMMERCE SYSTEMS

Authenticity: Participants in transactions cannot be impersonated and signatures cannot be forged.

Integrity: Documents (purchase orders, payment instructions,...) cannot be forged.

Privacy: Details of transaction should be kept secret.

Security: Sensitive information (as credit card numbers) must be protected.

Anonymity: Anonymity of money senders should be guaranteed.

BASIC REQUIREMENTS for e-COMMERCE SYSTEMS

Authenticity: Participants in transactions cannot be impersonated and signatures cannot be forged.

Integrity: Documents (purchase orders, payment instructions,...) cannot be forged.

Privacy: Details of transaction should be kept secret.

Security: Sensitive information (as credit card numbers) must be protected.

Anonymity: Anonymity of money senders should be guaranteed.

Additional requirement: In order to allow an efficient fighting of the organized crime a system for processing e-money has to be such that under well defined conditions it has to be possible to revoke customer's identity and flow

HISTORICAL COMMENT

So called Secure Electronic Transaction protocol was created to standardize the exchange of credit card information.

Development of SET initiated in 1996 credit card companies MasterCard and Visa.

EXAMPLE – DUAL SIGNATURE PROTOCOL

We present a protocol to solve the following security and privacy problem in e-commerce: How to arrange e-shopping in such a way that shoppers' banks should not know what shoppers/cardholders are ordering and shops should not learn credit card numbers of shoppers.

EXAMPLE – DUAL SIGNATURE PROTOCOL

We present a protocol to solve the following security and privacy problem in e-commerce: How to arrange e-shopping in such a way that shoppers' banks should not know what shoppers/cardholders are ordering and shops should not learn credit card numbers of shoppers.

Participants of our e-commerce protocol will be: a bank, a shopper/cardholder, a shop

EXAMPLE – DUAL SIGNATURE PROTOCOL

We present a protocol to solve the following security and privacy problem in e-commerce: How to arrange e-shopping in such a way that shoppers' banks should not know what shoppers/cardholders are ordering and shops should not learn credit card numbers of shoppers.

Participants of our e-commerce protocol will be: a bank, a shopper/cardholder, a shop

The cardholder will use the following information:

- GSO – Goods and Services Order (cardholder's name, shop's name, items being ordered, their quantity,...)
- PI - Payment Instructions (shop's name, card number, total price,...)

Protocol will use also a public hash function h.

EXAMPLE – DUAL SIGNATURE PROTOCOL

We present a protocol to solve the following security and privacy problem in e-commerce: How to arrange e-shopping in such a way that shoppers' banks should not know what shoppers/cardholders are ordering and shops should not learn credit card numbers of shoppers.

Participants of our e-commerce protocol will be: a bank, a shopper/cardholder, a shop

The cardholder will use the following information:

- GSO – Goods and Services Order (cardholder's name, shop's name, items being ordered, their quantity,...)
- PI - Payment Instructions (shop's name, card number, total price,...)

Protocol will use also a public hash function h.

RSA cryptosystem will also be used and

- e_C, e_S and e_B will be public (encryption) keys of cardholder, shop, bank and
- d_C, d_S and d_B will be their secret (decryption) keys.

CARDHOLDER and SHOP ACTIONS

A cardholder performs the following procedure – to create GSO-goods and services order

1. Computes $HEGSO = h(e_S(GSO))$ – hash value of the encryption of GSO.

CARDHOLDER and SHOP ACTIONS

A cardholder performs the following procedure – to create GSO-goods and services order

1. Computes $HEGSO = h(e_S(GSO))$ – hash value of the encryption of GSO.
2. Computes $HEPI = h(e_B(PI))$ – hash value of the encryption of the payment instructions for the bank.

A cardholder performs the following procedure – to create GSO-goods and services order

1. Computes $HEGSO = h(e_S(GSO))$ – hash value of the encryption of GSO.
2. Computes $HEPI = h(e_B(PI))$ – hash value of the encryption of the payment instructions for the bank.
3. Computes $HPO = h(HEPI\|HEGSO)$ – Hash value of the Payment Order.

CARDHOLDER and SHOP ACTIONS

A cardholder performs the following procedure – to create GSO-goods and services order

1. Computes $HEGSO = h(e_S(GSO))$ – hash value of the encryption of GSO.
2. Computes $HEPI = h(e_B(PI))$ – hash value of the encryption of the payment instructions for the bank.
3. Computes $HPO = h(HEPI \| HEGSO)$ – Hash value of the Payment Order.
4. Signs HPO by computing "Dual Signature" $DS = d_C(HPO)$.

CARDHOLDER and SHOP ACTIONS

A cardholder performs the following procedure – to create GSO-goods and services order

1. Computes $HEGSO = h(e_S(GSO))$ – hash value of the encryption of GSO.
2. Computes $HEPI = h(e_B(PI))$ – hash value of the encryption of the payment instructions for the bank.
3. Computes $HPO = h(HEPI \| HEGSO)$ – Hash value of the Payment Order.
4. Signs HPO by computing "Dual Signature" $DS = d_C(HPO)$.
5. Sends $e_S(GSO)$, DS, HEPI, and $e_B(PI)$ to the shop.

A cardholder performs the following procedure – to create GSO-goods and services order

1. Computes $HEGSO = h(e_S(GSO))$ – hash value of the encryption of GSO.
2. Computes $HEPI = h(e_B(PI))$ – hash value of the encryption of the payment instructions for the bank.
3. Computes $HPO = h(HEPI \| HEGSO)$ – Hash value of the Payment Order.
4. Signs HPO by computing "Dual Signature" $DS = d_C(HPO)$.
5. Sends $e_S(GSO)$, DS, HEPI, and $e_B(PI)$ to the shop.

The Shop does the following: – to create payment instructions

CARDHOLDER and SHOP ACTIONS

A cardholder performs the following procedure – to create GSO-goods and services order

1. Computes $HEGSO = h(e_S(GSO))$ – hash value of the encryption of GSO.
2. Computes $HEPI = h(e_B(PI))$ – hash value of the encryption of the payment instructions for the bank.
3. Computes $HPO = h(HEPI \| HEGSO)$ – Hash value of the Payment Order.
4. Signs HPO by computing "Dual Signature" $DS = d_C(HPO)$.
5. Sends $e_S(GSO)$, DS, HEPI, and $e_B(PI)$ to the shop.

The Shop does the following: – to create payment instructions

- Calculates $h(e_S(GSO)) =$ HEGSO;

CARDHOLDER and SHOP ACTIONS

A cardholder performs the following procedure – to create GSO-goods and services order

1. Computes $HEGSO = h(e_S(GSO))$ – hash value of the encryption of GSO.
2. Computes $HEPI = h(e_B(PI))$ – hash value of the encryption of the payment instructions for the bank.
3. Computes $HPO = h(HEPI \| HEGSO)$ – Hash value of the Payment Order.
4. Signs HPO by computing "Dual Signature" $DS = d_C(HPO)$.
5. Sends $e_S(GSO)$, DS, HEPI, and $e_B(PI)$ to the shop.

The Shop does the following: – to create payment instructions

- Calculates $h(e_S(GSO)) =$ HEGSO;
- Calculates $h(HEPI|HEGSO)$ and $e_C(DS)$. If they are equal, the shop has verified by that the cardholder signature;

CARDHOLDER and SHOP ACTIONS

A cardholder performs the following procedure – to create GSO-goods and services order

1. Computes $HEGSO = h(e_S(GSO))$ – hash value of the encryption of GSO.
2. Computes $HEPI = h(e_B(PI))$ – hash value of the encryption of the payment instructions for the bank.
3. Computes $HPO = h(HEPI \| HEGSO)$ – Hash value of the Payment Order.
4. Signs HPO by computing "Dual Signature" $DS = d_C(HPO)$.
5. Sends $e_S(GSO)$, DS, HEPI, and $e_B(PI)$ to the shop.

The Shop does the following: – to create payment instructions

- Calculates $h(e_S(GSO))$ = HEGSO;
- Calculates $h(HEPI|HEGSO)$ and $e_C(DS)$. If they are equal, the shop has verified by that the cardholder signature;
- Computes $d_S(e_S(GSO))$ to get GSO.

CARDHOLDER and SHOP ACTIONS

A cardholder performs the following procedure – to create GSO-goods and services order

1. Computes $HEGSO = h(e_S(GSO))$ – hash value of the encryption of GSO.
2. Computes $HEPI = h(e_B(PI))$ – hash value of the encryption of the payment instructions for the bank.
3. Computes $HPO = h(HEPI \| HEGSO)$ – Hash value of the Payment Order.
4. Signs HPO by computing "Dual Signature" $DS = d_C(HPO)$.
5. Sends $e_S(GSO)$, DS, HEPI, and $e_B(PI)$ to the shop.

The Shop does the following: – to create payment instructions

- Calculates $h(e_S(GSO)) =$ HEGSO;
- Calculates $h(HEPI|HEGSO)$ and $e_C(DS)$. If they are equal, the shop has verified by that the cardholder signature;
- Computes $d_S(e_S(GSO))$ to get GSO.
- Sends $HEGSO, HEPI, e_B(PI)$, and DS to the bank.

The Bank has received HEPI, HEGSO, $e_B(PI)$, and DS and performs the following actions.

1. Computes $h(e_B(PI))$ – which should be equal to HEPI.

BANK and SHOP ACTIONS

The Bank has received HEPI, HEGSO, $e_B(PI)$, and DS and performs the following actions.

1. Computes $h(e_B(PI))$ – which should be equal to HEPI.
2. Computes $h(h(e_B(PI)) \| HEGSO)$ which should be equal to $e_C(DS) = HPO$.

BANK and SHOP ACTIONS

The Bank has received HEPI, HEGSO, $e_B(PI)$, and DS and performs the following actions.

1. Computes $h(e_B(PI))$ – which should be equal to HEPI.
2. Computes $h(h(e_B(PI))\|HEGSO)$ which should be equal to $e_C(DS) = HPO$.
3. Computes $d_B(e_B(PI))$ to obtain PI;

BANK and SHOP ACTIONS

The Bank has received HEPI, HEGSO, $e_B(PI)$, and DS and performs the following actions.

1. Computes $h(e_B(PI))$ – which should be equal to HEPI.
2. Computes $h(h(e_B(PI))\|HEGSO)$ which should be equal to $e_C(DS) = HPO$.
3. Computes $d_B(e_B(PI))$ to obtain PI;
4. Returns an encrypted (with e_S) digitally signed authorization to shop, guaranteeing the payment.

BANK and SHOP ACTIONS

The Bank has received HEPI, HEGSO, $e_B(PI)$, and DS and performs the following actions.

1. Computes $h(e_B(PI))$ – which should be equal to HEPI.
2. Computes $h(h(e_B(PI))\|HEGSO)$ which should be equal to $e_C(DS) = HPO$.
3. Computes $d_B(e_B(PI))$ to obtain PI;
4. Returns an encrypted (with e_S) digitally signed authorization to shop, guaranteeing the payment.

Shop completes the procedure by encrypting, with e_C, the receipt to the cardholder, indicating that transaction has been completed.

BANK and SHOP ACTIONS

The Bank has received HEPI, HEGSO, $e_B(PI)$, and DS and performs the following actions.

1. Computes $h(e_B(PI))$ – which should be equal to HEPI.
2. Computes $h(h(e_B(PI))\|HEGSO)$ which should be equal to $e_C(DS) = HPO$.
3. Computes $d_B(e_B(PI))$ to obtain PI;
4. Returns an encrypted (with e_S) digitally signed authorization to shop, guaranteeing the payment.

Shop completes the procedure by encrypting, with e_C, the receipt to the cardholder, indicating that transaction has been completed.

It is easy to verify that the above protocol fulfills basic requirements concerning security, privacy and integrity.

DIGITAL MONEY

Is it possible to have electronic (digital) money?

DIGITAL MONEY

Is it possible to have electronic (digital) money?

It seems that not, because copies of digital information are indistinguishable from their origin and one could therefore hardly prevent double spending,....

DIGITAL MONEY

Is it possible to have electronic (digital) money?

It seems that not, because copies of digital information are indistinguishable from their origin and one could therefore hardly prevent double spending,....

T. Okamoto and K. Ohia formulated six properties digital money systems should have.

DIGITAL MONEY

Is it possible to have electronic (digital) money?

It seems that not, because copies of digital information are indistinguishable from their origin and one could therefore hardly prevent double spending,....

T. Okamoto and K. Ohia formulated six properties digital money systems should have.

1. One should be able to send e-money through e-networks.

DIGITAL MONEY

Is it possible to have electronic (digital) money?

It seems that not, because copies of digital information are indistinguishable from their origin and one could therefore hardly prevent double spending,....

T. Okamoto and K. Ohia formulated six properties digital money systems should have.

1. One should be able to send e-money through e-networks.
2. It should not be possible to copy and reuse e-money.

DIGITAL MONEY

Is it possible to have electronic (digital) money?

It seems that not, because copies of digital information are indistinguishable from their origin and one could therefore hardly prevent double spending,....

T. Okamoto and K. Ohia formulated six properties digital money systems should have.

1. One should be able to send e-money through e-networks.
2. It should not be possible to copy and reuse e-money.
3. Transactions using e-money could be done **off-line** – that is no communication with central bank should be needed during translation.

DIGITAL MONEY

Is it possible to have electronic (digital) money?

It seems that not, because copies of digital information are indistinguishable from their origin and one could therefore hardly prevent double spending,....

T. Okamoto and K. Ohia formulated six properties digital money systems should have.

1. One should be able to send e-money through e-networks.
2. It should not be possible to copy and reuse e-money.
3. Transactions using e-money could be done **off-line** – that is no communication with central bank should be needed during translation.
4. One should be able to sent e-money to anybody.

DIGITAL MONEY

Is it possible to have electronic (digital) money?

It seems that not, because copies of digital information are indistinguishable from their origin and one could therefore hardly prevent double spending,....

T. Okamoto and K. Ohia formulated six properties digital money systems should have.

1. One should be able to send e-money through e-networks.
2. It should not be possible to copy and reuse e-money.
3. Transactions using e-money could be done **off-line** – that is no communication with central bank should be needed during translation.
4. One should be able to sent e-money to anybody.
5. An e-coin could be divided into e-coins of smaller values.

Several systems of e-money have been created that satisfy all or at least some of the above requirements.

BLIND SIGNATURES – APPLICATIONS

Blind digital signatures allow the signer (bank) to sign a message without seeing its content.

Blind digital signatures allow the signer (bank) to sign a message without seeing its content.

Scenario: Customer Bob would like to give e-money to Shop. E-money has to be signed by a Bank. Shop must be able to verify Bank's signature. Later, when Shop sends e-money to Bank, Bank should not be able to recognize that it signed these e-money for Bob. Bank has therefore to sign money blindly.

Bob can obtain a blind signature for a message m from Bank by executing the Schnorr blind signature protocol described on the next slide.

Blind digital signatures allow the signer (bank) to sign a message without seeing its content.

Scenario: Customer Bob would like to give e-money to Shop. E-money has to be signed by a Bank. Shop must be able to verify Bank's signature. Later, when Shop sends e-money to Bank, Bank should not be able to recognize that it signed these e-money for Bob. Bank has therefore to sign money blindly.

Bob can obtain a blind signature for a message m from Bank by executing the Schnorr blind signature protocol described on the next slide.

Basic setting

Bank chooses large primes $p, q|(p-1)$ and an $g \in Z_p$ of order q.
Let $h : \{0,1\}^* \rightarrow Z_p$ be a collision-free hash function.
Bank's secret will be a randomly chosen $x \in \{0, \ldots, p-1\}$.
Public information: $(p, q, g, y = g^x)$.

1. Schnorr's simplified identification scheme in which Bank proves its identity by proving that it knows x.
   - Bank chooses a random $r \in \{0, \dots, q-1\}$ and send $a = g^r$ to Bob. {By that Bank "commits" itself to r}.
   - Bob sends to Bank a random $c \in \{0, \dots, q-1\}$ {a challenge}.
   - Bank sends to Bob $b = r - cx$ {a response}.
   - Bob accepts the proof that bank knows x if $a = g^b y^c$. {because $y = g^x$}

BLIND SIGNATURES – protocols

1. Schnorr's simplified identification scheme in which Bank proves its identity by proving that it knows x.
   - Bank chooses a random $r \in \{0, \dots, q-1\}$ and send $a = g^r$ to Bob. {By that Bank "commits" itself to r}.
   - Bob sends to Bank a random $c \in \{0, \dots, q-1\}$ {a challenge}.
   - Bank sends to Bob b = r − cx {a response}.
   - Bob accepts the proof that bank knows x if $a = g^b y^c$. {because $y = g^x$}
2. Transfer of the identification scheme to a signature scheme:

   Bob chooses as $c = h(m \| a)$, where m is message to sign.

   Signature: (c, b); Verification rule: $a = g^b y^c$; Transcript: (a, c, b).

1. Schnorr's simplified identification scheme in which Bank proves its identity by proving that it knows x.
   - Bank chooses a random $r \in \{0, \ldots, q-1\}$ and send $a = g^r$ to Bob. {By that Bank "commits" itself to r}.
   - Bob sends to Bank a random $c \in \{0, \ldots, q-1\}$ {a challenge}.
   - Bank sends to Bob b = r − cx {a response}.
   - Bob accepts the proof that bank knows x if $a = g^b y^c$. {because $y = g^x$}
2. Transfer of the identification scheme to a signature scheme:

   Bob chooses as $c = h(m\|a)$, where m is message to sign.

   Signature: (c, b); Verification rule: $a = g^b y^c$; Transcript: (a, c, b).
3. **Shnorr's blind signature scheme**
   - Bank sends to Bob $a' = g^{r'}$ with random $r' \in \{0, \ldots, q-1\}$.
   - Bob chooses random $u, v, w \in \{0, \ldots, q-1\}$, $u \neq 0$, computes $a = a'^u g^v y^w$, $c = h(m\|a)$, $c' = (c-w)u^{-1}$ and sends c' to Bank.
   - Bank sends to Bob b' = r' - c'x.

   Bob verifies whether $a' = g^{b'} y^{c'}$, computes b = ub' + v and gets blind signature σ(m) = (c, b) of m.

   Verification condition for the blind signature: $c = h(m\|g^b y^c)$.

   Both (a,c,b) and (a',c',b') are valid transcripts.

COMPUTATION of DECEPTION PROBABILITIES I

Probability of impersonation: For $w \in M, t \in T$, let us define payoff(w, t) to be the probability that Bob accepts the message (w, t) as authentic. Then

$$payoff(w,t) = Pr(t = a_{k_0}(w)) \qquad (4)$$
$$= \sum_{\{k \in K | a_k(w) = t\}} Pr_K(k) \qquad (5)$$

In other words, payoff(w, t) is computed by selecting the rows of the authentication matrix that have entry t in column w and summing probabilities of the corresponding keys.

Therefore $\quad P_i = max\{payoff(w,t), | w \in M, t \in A\}$.

COMPUTATION of DECEPTION PROBABILITIES I

Probability of impersonation: For $w \in M, t \in T$, let us define payoff(w, t) to be the probability that Bob accepts the message (w, t) as authentic. Then

$$payoff(w,t) = Pr(t = a_{k_0}(w)) \qquad (4)$$
$$= \sum_{\{k \in K | a_k(w) = t\}} Pr_K(k) \qquad (5)$$

In other words, payoff(w, t) is computed by selecting the rows of the authentication matrix that have entry t in column w and summing probabilities of the corresponding keys.

Therefore $\quad P_i = max\{payoff(w,t), | w \in M, t \in A\}$.

Probability of substitution: Define, for $w, w' \in M, w \neq w'$ and $t, t' \in A, payoff(w', t', w, t)$ to be the probability that a substitution of (w, t) with (w', t') will succeed to deceive Bob. Hence

$$payoff(w', t', w, t) = Pr(t' = a_{k_0}(w') | t = a_{k_0}(w)) \qquad (6)$$
$$= \frac{Pr(t' = a_{k_0}(w') \cap t = e_{k_0}(w))}{Pr(t = a_{k_0}(w))} \qquad (7)$$
$$= \frac{\sum_{\{k \in K | a_k(w) = t, a_k(w') = t'\}} p_k(k)}{payoff(w,t)} \qquad (8)$$

Observe that the numerator in the last fraction is found by selecting rows of the authentication matrix with value t in column w and t' in column w'.

COMPUTATION of DECEPTION PROBABILITIES II

Since Mallot wants to maximize his chance of deceiving Bob, he needs to compute

$$p_{w,t} = max\{payoff(w', t', w, t) | w' \in M, w \neq w', t' \in A\}.$$

$p_{w,t}$ therefore denotes the probability that Mallot can deceive Bob with a substitution in the case (w, t) is the message observed.

If $Pr_{\mathrm{Ma}}(w, t)$ is the probability of observing a message (w, t) in the channel, then

$$P_S = \sum_{(w,t) \in Ma} Pr_{Ma}(w, t) p_{w,t}$$

and

$$Pr_{Ma}(w, t) = Pr_M(w) Pr_K(t|w) = Pr_M(w) \times payoff(w, t).$$

The next problem is to show how to construct an authentication code such that the deception probabilities are as low as possible.

The concept of orthogonal arrays, introduced next, serves well such a purpose.

Part X

Protocols to do seemingly impossible and zero-knowledge protocols

PROTOCOLS to do SEEMINGLY IMPOSSIBLE

A **protocol** is an algorithm two (or more) parties have to follow to perform a communication/cooperation.

A cryptographical protocol is a protocol to achieve secure communication during some goal oriented cooperation.

PROTOCOLS to do SEEMINGLY IMPOSSIBLE

A **protocol** is an algorithm two (or more) parties have to follow to perform a communication/cooperation.

A cryptographical protocol is a protocol to achieve secure communication during some goal oriented cooperation.

In this chapter we first present several cryptographic protocols for such basic cryptographic primitives as coin tossing, bit commitment and oblivious transfer.

PROTOCOLS to do SEEMINGLY IMPOSSIBLE

A **protocol** is an algorithm two (or more) parties have to follow to perform a communication/cooperation.

A cryptographical protocol is a protocol to achieve secure communication during some goal oriented cooperation.

In this chapter we first present several cryptographic protocols for such basic cryptographic primitives as coin tossing, bit commitment and oblivious transfer.

After that we deal with a variety of cryptographical protocols that allow to solve easily seemingly unsolvable problems.

A **protocol** is an algorithm two (or more) parties have to follow to perform a communication/cooperation.

A cryptographical protocol is a protocol to achieve secure communication during some goal oriented cooperation.

In this chapter we first present several cryptographic protocols for such basic cryptographic primitives as coin tossing, bit commitment and oblivious transfer.

After that we deal with a variety of cryptographical protocols that allow to solve easily seemingly unsolvable problems.

Of special importance among them are so called zero-knowledge protocols we will deal with afterwards. They are counter intuitive, though powerful and useful.

PRIMITIVES for CRYPTOGRAPHIC PROTOCOLS

Cryptographic protocols are specifications how two parties, Alice and Bob, should prepare themselves for a communication and how they should behave during a communication in order to achieve their goal and be protected against an adversary.

PRIMITIVES for CRYPTOGRAPHIC PROTOCOLS

Cryptographic protocols are specifications how two parties, Alice and Bob, should prepare themselves for a communication and how they should behave during a communication in order to achieve their goal and be protected against an adversary.

In **coin-flipping protocols** Alice and Bob can flip a coin over a distance in such a way that neither of them can determine the outcome of the flip, but both can agree on the outcome in spite of the fact that they do not trust each other.

PRIMITIVES for CRYPTOGRAPHIC PROTOCOLS

Cryptographic protocols are specifications how two parties, Alice and Bob, should prepare themselves for a communication and how they should behave during a communication in order to achieve their goal and be protected against an adversary.

In **coin-flipping protocols** Alice and Bob can flip a coin over a distance in such a way that neither of them can determine the outcome of the flip, but both can agree on the outcome in spite of the fact that they do not trust each other.

In **bit commitment protocols** Alice can choose a bit and get committed to it in the following sense: Bob has no way of learning Alice's commitment and Alice has no way of changing her commitment. Alice commits herself to a bit x using a *commit*(x) procedure, and reveals her commitment, if needed, using *open*(x) procedure.

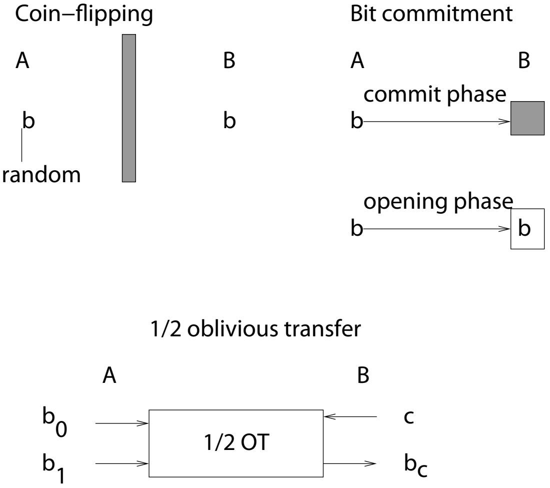
PRIMITIVES for CRYPTOGRAPHIC PROTOCOLS

Cryptographic protocols are specifications how two parties, Alice and Bob, should prepare themselves for a communication and how they should behave during a communication in order to achieve their goal and be protected against an adversary.

In **coin-flipping protocols** Alice and Bob can flip a coin over a distance in such a way that neither of them can determine the outcome of the flip, but both can agree on the outcome in spite of the fact that they do not trust each other.

In **bit commitment protocols** Alice can choose a bit and get committed to it in the following sense: Bob has no way of learning Alice's commitment and Alice has no way of changing her commitment. Alice commits herself to a bit x using a *commit*(x) procedure, and reveals her commitment, if needed, using *open*(x) procedure. In 1-**out**-2

oblivious transfer protocols Alice transmits two messages m_1 and m_2 to Bob who can chose whether to receive m_1 or m_2, but cannot learn both, and Alice has no idea which of them Bob has received.


Coin–flipping
A
B
b
b
random
Bit commitment
A
B
commit phase
b
opening phase
b
b
1/2 oblivious transfer
A
B
b0
b1
1/2 OT
c
bc

PROTOCOLS for COIN-FLIPPING BY PHONE

Coin-flipping by telephone:

Alice and Bob got divorced and they do not trust each other any longer. They want to decide, communicating by phone only, who gets the car.

PROTOCOLS for COIN-FLIPPING BY PHONE

Coin-flipping by telephone:

Alice and Bob got divorced and they do not trust each other any longer. They want to decide, communicating by phone only, who gets the car.

Protocol 1 Alice sends Bob messages head and tail encrypted by a one-way function f. Bob guesses which one of them is encryption of head. Alice tells Bob whether his guess was correct. If Bob does not believe her, Alice sends f to Bob.

PROTOCOLS for COIN-FLIPPING BY PHONE

Coin-flipping by telephone:

Alice and Bob got divorced and they do not trust each other any longer. They want to decide, communicating by phone only, who gets the car.

Protocol 1 Alice sends Bob messages head and tail encrypted by a one-way function f. Bob guesses which one of them is encryption of head. Alice tells Bob whether his guess was correct. If Bob does not believe her, Alice sends f to Bob.

Protocol 2 Alice chooses two large primes p,q, sends Bob n = pq and keeps p, q secret.

PROTOCOLS for COIN-FLIPPING BY PHONE

Coin-flipping by telephone:

Alice and Bob got divorced and they do not trust each other any longer. They want to decide, communicating by phone only, who gets the car.

Protocol 1 Alice sends Bob messages head and tail encrypted by a one-way function f. Bob guesses which one of them is encryption of head. Alice tells Bob whether his guess was correct. If Bob does not believe her, Alice sends f to Bob.

Protocol 2 Alice chooses two large primes p,q, sends Bob n = pq and keeps p, q secret. Bob chooses randomly an integer $y \in \{1, \ldots, \frac{n}{2}\}$, sends Alice $x = y^2 \mod n$ and tells Alice: if you guess y correctly, car will be yours.

PROTOCOLS for COIN-FLIPPING BY PHONE

Coin-flipping by telephone:

Alice and Bob got divorced and they do not trust each other any longer. They want to decide, communicating by phone only, who gets the car.

Protocol 1 Alice sends Bob messages head and tail encrypted by a one-way function f. Bob guesses which one of them is encryption of head. Alice tells Bob whether his guess was correct. If Bob does not believe her, Alice sends f to Bob.

Protocol 2 Alice chooses two large primes p,q, sends Bob n = pq and keeps p, q secret. Bob chooses randomly an integer $y \in \{1, \dots, \frac{n}{2}\}$, sends Alice $x = y^2 \mod n$ and tells Alice: if you guess y correctly, car will be yours.

Alice computes four square roots $(x_1, n - x_1)$ and $(x_2, n - x_2)$ of x.

Let

$$x_1' = min(x_1, n - x_1), x_2' = min(x_2, n - x_2).$$

Since $y \in \{1, \dots, \frac{n}{2}\}$, either $y = x_1'$ or $y = x_2'$.

Alice then guesses whether $y = x_1'$ or $y = x_2'$ and tells Bob her choice (for example by reporting the position and value of the leftmost bit in which x_1' and x_2' differ).

PROTOCOLS for COIN-FLIPPING BY PHONE

Coin-flipping by telephone:

Alice and Bob got divorced and they do not trust each other any longer. They want to decide, communicating by phone only, who gets the car.

Protocol 1 Alice sends Bob messages head and tail encrypted by a one-way function f. Bob guesses which one of them is encryption of head. Alice tells Bob whether his guess was correct. If Bob does not believe her, Alice sends f to Bob.

Protocol 2 Alice chooses two large primes p,q, sends Bob n = pq and keeps p, q secret. Bob chooses randomly an integer $y \in \{1, \ldots, \frac{n}{2}\}$, sends Alice $x = y^2 \mod n$ and tells Alice: if you guess y correctly, car will be yours.

Alice computes four square roots $(x_1, n - x_1)$ and $(x_2, n - x_2)$ of x.

Let

$$x_1' = min(x_1, n - x_1), x_2' = min(x_2, n - x_2).$$

Since $y \in \{1, \ldots, \frac{n}{2}\}$, either $y = x_1'$ or $y = x_2'$.

Alice then guesses whether $y = x_1'$ or $y = x_2'$ and tells Bob her choice (for example by reporting the position and value of the leftmost bit in which x_1' and x_2' differ).

Bob tells Alice whether her guess was correct.

(Later, if necessary, Alice reveals p and q, and Bob reveals y.)

COIN TOSSING – requirements and problems

- In any good coin tossing protocol both parties should influence the outcome and should accept the outcome. Both outcomes should have the same probability.
- Requirements for a coin tossing protocol are sometimes generalized as follows:

COIN TOSSING – requirements and problems

- In any good coin tossing protocol both parties should influence the outcome and should accept the outcome. Both outcomes should have the same probability.
- Requirements for a coin tossing protocol are sometimes generalized as follows:
  - The outcome of the protocol is an element from the set {0, 1, reject}
  - If both parties behave correctly, the outcome should be from the set {0, 1}
  - If it is not the case that both parties behave correctly, the outcome should be reject

COIN TOSSING – requirements and problems

- In any good coin tossing protocol both parties should influence the outcome and should accept the outcome. Both outcomes should have the same probability.
- Requirements for a coin tossing protocol are sometimes generalized as follows:
  - The outcome of the protocol is an element from the set {0, 1, reject}
  - If both parties behave correctly, the outcome should be from the set {0, 1}
  - If it is not the case that both parties behave correctly, the outcome should be reject

Problem: In some coin tossing protocols one party can find out the outcome sooner than the second party. In such a case if she is not happy with the outcome she can disrupt the protocol – to produce reject or to say "I do not continue in performing the protocol". A way out is to require that in case of correct behavior no outcome should have probability $> \frac{1}{2}$.

COIN TOSSING USING a ONE-WAY FUNCTION

Protocol:

- **Alice chooses a one-way function f and informs Bob about the definition domain of f.**

COIN TOSSING USING a ONE-WAY FUNCTION

Protocol:

- **Alice chooses a one-way function f and informs Bob about the definition domain of f.**
- Bob chooses randomly r_1, r_2 from dom(f) and sends them to Alice

COIN TOSSING USING a ONE-WAY FUNCTION

Protocol:

- **Alice chooses a one-way function f and informs Bob about the definition domain of f.**
- Bob chooses randomly r_1, r_2 from dom(f) and sends them to Alice
- Alice sends to Bob one of the values $f(r_1)$ or $f(r_2)$

Protocol:

- **Alice chooses a one-way function f and informs Bob about the definition domain of f**.
- Bob chooses randomly r_1, r_2 from dom(f) and sends them to Alice
- Alice sends to Bob one of the values $f(r_1)$ or $f(r_2)$
- Bob announces Alice his guess which of the two values he received

COIN TOSSING USING a ONE-WAY FUNCTION

Protocol:

- **Alice chooses a one-way function f and informs Bob about the definition domain of f**.
- Bob chooses randomly r_1, r_2 from dom(f) and sends them to Alice
- Alice sends to Bob one of the values $f(r_1)$ or $f(r_2)$
- Bob announces Alice his guess which of the two values he received
- Alice announces Bob whether his guess was correct (0) or not (1)

COIN TOSSING USING a ONE-WAY FUNCTION

Protocol:

- **Alice chooses a one-way function f and informs Bob about the definition domain of f**.
- Bob chooses randomly r_1, r_2 from dom(f) and sends them to Alice
- Alice sends to Bob one of the values $f(r_1)$ or $f(r_2)$
- Bob announces Alice his guess which of the two values he received
- Alice announces Bob whether his guess was correct (0) or not (1)
- If one needs to verify correctness, Alice should send to Bob specification of f

COIN TOSSING USING a ONE-WAY FUNCTION

Protocol:

- **Alice chooses a one-way function f and informs Bob about the definition domain of f**.
- Bob chooses randomly r_1, r_2 from dom(f) and sends them to Alice
- Alice sends to Bob one of the values $f(r_1)$ or $f(r_2)$
- Bob announces Alice his guess which of the two values he received
- Alice announces Bob whether his guess was correct (0) or not (1)
- If one needs to verify correctness, Alice should send to Bob specification of f

The protocol is computationally secure. Indeed, to cheat, Alice should be able to find, for randomly chosen r_1, r_2 such a one-way function f that $f(r_1) = f(r_2)$.

Basic ideas and solutions I

In a bit commitment protocol Alice chooses a bit b and gets committed to b, in the following sense:

Basic ideas and solutions I

In a bit commitment protocol Alice chooses a bit b and gets committed to b, in the following sense:

Bob has no way of knowing which commitment Alice has made, and Alice has no way of changing her commitment once she has made it; say after Bob announces his guess as to what Alice has chosen.

Basic ideas and solutions I

In a bit commitment protocol Alice chooses a bit b and gets committed to b, in the following sense:

Bob has no way of knowing which commitment Alice has made, and Alice has no way of changing her commitment once she has made it; say after Bob announces his guess as to what Alice has chosen.

An example of a "pre-computer era" bit commitment protocol is that Alice writes her commitment on a paper, locks it in a box, sends the box to Bob and, later, in the opening phase, she sends also the key to Bob.

Basic ideas and solutions I

In a bit commitment protocol Alice chooses a bit b and gets committed to b, in the following sense:

Bob has no way of knowing which commitment Alice has made, and Alice has no way of changing her commitment once she has made it; say after Bob announces his guess as to what Alice has chosen.

An example of a "pre-computer era" bit commitment protocol is that Alice writes her commitment on a paper, locks it in a box, sends the box to Bob and, later, in the opening phase, she sends also the key to Bob.

Complexity era solution I. Alice chooses a one-way function f and an even (odd) x if she wants to commit herself to 0 (1) and sends to Bob f(x) and f.

Problem: Alice may know an even x_1 and an odd x_2 such that $f(x_1) = f(x_2)$.

Basic ideas and solutions I

In a bit commitment protocol Alice chooses a bit b and gets committed to b, in the following sense:

Bob has no way of knowing which commitment Alice has made, and Alice has no way of changing her commitment once she has made it; say after Bob announces his guess as to what Alice has chosen.

An example of a "pre-computer era" bit commitment protocol is that Alice writes her commitment on a paper, locks it in a box, sends the box to Bob and, later, in the opening phase, she sends also the key to Bob.

Complexity era solution I. Alice chooses a one-way function f and an even (odd) x if she wants to commit herself to 0 (1) and sends to Bob f(x) and f.

Problem: Alice may know an even x_1 and an odd x_2 such that $f(x_1) = f(x_2)$.

Complexity era solution II. Alice chooses a one-way function f, two random x_1, x_2 and a bit b she wishes to commit to, and sends to Bob $(f(x_1, x_2, b), x_1)$ - a commitment.

When times comes for Alice to reveal her bit she sends to Bob f and the triple (x_1, x_2, b).

BIT COMMITMENT SCHEMES I

The basis of bit commitment protocols are bit commitment schemes:

BIT COMMITMENT SCHEMES I

The basis of bit commitment protocols are bit commitment schemes:

A bit commitment scheme is a mapping $f : \{0, 1\} \times X \rightarrow Y$, where X and Y are finite sets.

BIT COMMITMENT SCHEMES I

The basis of bit commitment protocols are bit commitment schemes:

A bit commitment scheme is a mapping $f : \{0,1\} \times X \rightarrow Y$, where X and Y are finite sets.

A commitment to a $b \in \{0,1\}$, or an encryption of b, is any value (called a blow) f(b, x) where $x \in X$.

BIT COMMITMENT SCHEMES I

The basis of bit commitment protocols are bit commitment schemes:

A bit commitment scheme is a mapping $f : \{0,1\} \times X \to Y$, where X and Y are finite sets.

A commitment to a $b \in \{0,1\}$, or an encryption of b, is any value (called a blow) f(b, x) where $x \in X$.

Each bit commitment protocol has two phases:

BIT COMMITMENT SCHEMES I

The basis of bit commitment protocols are bit commitment schemes:

A bit commitment scheme is a mapping $f : \{0,1\} \times X \to Y$, where X and Y are finite sets.

A commitment to a $b \in \{0,1\}$, or an encryption of b, is any value (called a blow) f(b, x) where x $\in$ X.

Each bit commitment protocol has two phases:

Commitment phase: The sender sends a bit b he wants to commit to, in an encrypted form, to the receiver.

BIT COMMITMENT SCHEMES I

The basis of bit commitment protocols are bit commitment schemes:

A bit commitment scheme is a mapping $f : \{0,1\} \times X \rightarrow Y$, where X and Y are finite sets.

A commitment to a $b \in \{0,1\}$, or an encryption of b, is any value (called a blow) f(b, x) where x $\in$ X.

Each bit commitment protocol has two phases:

Commitment phase: The sender sends a bit b he wants to commit to, in an encrypted form, to the receiver.

Opening phase: If required, the sender sends to the receiver additional information that enables the receiver to get b.

Each bit commitment scheme should have three properties:

Hiding (privacy): For no b $\in \{0,1\}$ and no $x \in X$, it is feasible for Bob to determine b from B = f(b, x).

BIT COMMITMENT SCHEMES II

Each bit commitment scheme should have three properties:

Hiding (privacy): For no b $\in \{0, 1\}$ and no $x \in X$, it is feasible for Bob to determine b from B = f(b, x).

Binding: Alice can "open" her commitment b, by revealing (opening) x and b such that B = f(b, x), but she should not be able to open a commitment (blow) B as both 0 and 1.

BIT COMMITMENT SCHEMES II

Each bit commitment scheme should have three properties:

Hiding (privacy): For no b $\in \{0, 1\}$ and no $x \in X$, it is feasible for Bob to determine b from B = f(b, x).

Binding: Alice can "open" her commitment b, by revealing (opening) x and b such that B = f(b, x), but she should not be able to open a commitment (blow) B as both 0 and 1.

Correctness: If both, the sender and the receiver, follow the protocol, then the receiver will always learn (recover) the committed value b.

BIT COMMITMENT with ONE-WAY FUNCTIONS

Commitment phase:

- Alice and Bob choose a one-way function f
- Bob sends a randomly chosen r_1 to Alice
- Alice chooses random r_2 and her committed bit b and sends to Bob $f(r_1, r_2, b)$.

BIT COMMITMENT with ONE-WAY FUNCTIONS

Commitment phase:

- Alice and Bob choose a one-way function f
- Bob sends a randomly chosen r_1 to Alice
- Alice chooses random r_2 and her committed bit b and sends to Bob $f(r_1, r_2, b)$.

Opening phase:

- Alice sends to Bob r_2 and b
- Bob computes $f(r_1, r_2, b)$ and compares with the value he has already received.

A commitment to a data w, without revealing w, using a hash function h, can be done as follows:

A commitment to a data w, without revealing w, using a hash function h, can be done as follows:

Commitment phase: To commit to a w choose a random r and make public h(wr).

A commitment to a data w, without revealing w, using a hash function h, can be done as follows:

Commitment phase: To commit to a w choose a random r and make public h(wr).

Opening phase: reveal r and w.

HASH FUNCTIONS and COMMITMENTS

A commitment to a data w, without revealing w, using a hash function h, can be done as follows:

Commitment phase: To commit to a w choose a random r and make public h(wr).

Opening phase: reveal r and w.

For this application the hash function h has to be one-way: from h(wr) it should be unfeasible to determine wr

TWO SPECIAL BIT COMMITMENT SCHEMES

Bit commitment scheme I. Let p, q be large primes, n = pq, m ∈ QNR(n), X = Y = Z_n^*. Let n,m be public.

Commitment: $f(b, x) = m^b x^2 \mod n$ for a random x from X.

Since computation of quadratic residues is in general infeasible, this bit commitment scheme is hiding.

Since $m \in QNR(n)$, there are no x_1, x_2 such that $mx_1^2 = x_2^2 \mod n$ and therefore the scheme is binding.

TWO SPECIAL BIT COMMITMENT SCHEMES

Bit commitment scheme I. Let p, q be large primes, $n = pq$, $m \in QNR(n)$, $X = Y = Z_n^*$. Let n,m be public.

Commitment: $f(b, x) = m^b x^2 \mod n$ for a random x from X.

Since computation of quadratic residues is in general infeasible, this bit commitment scheme is hiding.

Since $m \in QNR(n)$, there are no x_1, x_2 such that $mx_1^2 = x_2^2 \mod n$ and therefore the scheme is binding.

Bit commitment scheme II. Let p be a large Blum prime, $X = Z_p* = Y$, α be a primitive element of Z_p^*.

$$\begin{aligned} f(b, x) &= \alpha^x \mod p, \text{ if SLB(x) = b;} \\ &= \alpha^{p-x} \mod p, \text{ if SLB(x)} \neq \text{b.} \end{aligned}$$

TWO SPECIAL BIT COMMITMENT SCHEMES

Bit commitment scheme I. Let p, q be large primes, n = pq, m $\in$ QNR(n), X = Y = Z_n^*. Let n,m be public.

Commitment: $f(b, x) = m^b x^2 \mod n$ for a random x from X.

Since computation of quadratic residues is in general infeasible, this bit commitment scheme is hiding.

Since $m \in QNR(n)$, there are no x_1, x_2 such that $mx_1^2 = x_2^2 \mod n$ and therefore the scheme is binding.

Bit commitment scheme II. Let p be a large Blum prime, $X = Z_p* = Y$, α be a primitive element of Z_p^*.

$$\begin{aligned} f(b, x) &= \alpha^x \mod p, \text{ if SLB(x) = b;} \\ &= \alpha^{p-x} \mod p, \text{ if SLB(x)} \neq \text{b.} \end{aligned}$$

where

$$\begin{aligned} \text{SLB(x)} &= 0 \text{ if } x \equiv 0, 1 \text{ (mod 4);} \\ &= 1 \text{ if } x \equiv 2, 3 \text{ (mod 4).} \end{aligned}$$

TWO SPECIAL BIT COMMITMENT SCHEMES

Bit commitment scheme I. Let p, q be large primes, $n = pq$, $m \in QNR(n)$, $X = Y = Z_n^*$. Let n,m be public.

$$\text{Commitment: } f(b, x) = m^b x^2 \mod n \text{ for a random } x \text{ from } X.$$

Since computation of quadratic residues is in general infeasible, this bit commitment scheme is hiding.

Since $m \in QNR(n)$, there are no x_1, x_2 such that $mx_1^2 = x_2^2 \mod n$ and therefore the scheme is binding.

Bit commitment scheme II. Let p be a large Blum prime, $X = Z_p* = Y$, α be a primitive element of Z_p^*.

$$f(b,x) = \alpha^x \mod p, \text{ if } SLB(x) = b;$$
$$= \alpha^{p-x} \mod p, \text{ if } SLB(x) \neq b.$$

where

$$SLB(x) = 0 \text{ if } x \equiv 0, 1 \pmod 4;$$
$$= 1 \text{ if } x \equiv 2, 3 \pmod 4.$$

Binding property of this bit commitment scheme follows from the fact that in the case of discrete logarithms modulo Blum primes there is no effective way to determine second least significant bit (SLB) of the discrete logarithm.

MAKING COIN TOSSING FROM BIT COMMITMENT

Each bit commitment scheme can be used to solve coin tossing problem as follows:

1. Alice tosses a coin, and commits itself to its outcome b_A (say to 0 (1) if the outcome is head (tail)) and sends the commitment to Bob.

MAKING COIN TOSSING FROM BIT COMMITMENT

Each bit commitment scheme can be used to solve coin tossing problem as follows:

1. Alice tosses a coin, and commits itself to its outcome b_A (say to 0 (1) if the outcome is head (tail)) and sends the commitment to Bob.
2. Bob also tosses a coin and sends the outcome b_B to Alice.

MAKING COIN TOSSING FROM BIT COMMITMENT

Each bit commitment scheme can be used to solve coin tossing problem as follows:

1. Alice tosses a coin, and commits itself to its outcome b_A (say to 0 (1) if the outcome is head (tail)) and sends the commitment to Bob.
2. Bob also tosses a coin and sends the outcome b_B to Alice.
3. Alice opens her commitment. to Bob (so he knows b_A)

MAKING COIN TOSSING FROM BIT COMMITMENT

Each bit commitment scheme can be used to solve coin tossing problem as follows:

1. Alice tosses a coin, and commits itself to its outcome b_A (say to 0 (1) if the outcome is head (tail)) and sends the commitment to Bob.
2. Bob also tosses a coin and sends the outcome b_B to Alice.
3. Alice opens her commitment. to Bob (so he knows b_A)
4. Both Alice and Bob compute $b = b_A \oplus b_B$.

MAKING COIN TOSSING FROM BIT COMMITMENT

Each bit commitment scheme can be used to solve coin tossing problem as follows:

1. Alice tosses a coin, and commits itself to its outcome b_A (say to 0 (1) if the outcome is head (tail)) and sends the commitment to Bob.
2. Bob also tosses a coin and sends the outcome b_B to Alice.
3. Alice opens her commitment. to Bob (so he knows b_A)
4. Both Alice and Bob compute $b = b_A \oplus b_B$.

Observe that if at least one of the parties follows the protocol, that is it tosses a random coin, the outcome is indeed a random bit.

MAKING COIN TOSSING FROM BIT COMMITMENT

Each bit commitment scheme can be used to solve coin tossing problem as follows:

1. Alice tosses a coin, and commits itself to its outcome b_A (say to 0 (1) if the outcome is head (tail)) and sends the commitment to Bob.
2. Bob also tosses a coin and sends the outcome b_B to Alice.
3. Alice opens her commitment. to Bob (so he knows b_A)
4. Both Alice and Bob compute $b = b_A \oplus b_B$.

Observe that if at least one of the parties follows the protocol, that is it tosses a random coin, the outcome is indeed a random bit.

Note: Observe that after step 2 Alice will know what the outcome is, but Bob does not. So Alice can disrupt the protocol if the outcome is to be not good for her. This is a weak point of this protocol.

BASIC TYPES of HIDING and BINDING

If the hiding or the binding property of a commitment protocol depends on the complexity of a computational problem, we speak about computational hiding and computational binding.

In case, the binding or the hiding property does not depend on the complexity of a computational problem, we speak about unconditional hiding or unconditional binding.

A COMMITMENT SCHEME BASED on DISCRETE LOGARITHM

Alice wants to commit herself to an $m \in \{0, \ldots, q-1\}$.

Scheme setting:

A COMMITMENT SCHEME BASED on DISCRETE LOGARITHM

Alice wants to commit herself to an $m \in \{0, \ldots, q-1\}$.

Scheme setting:

Bob randomly chooses primes p and q such that

$$q | (p-1).$$

Bob chooses random generators $g \neq 1 \neq v$ of the subgroup G of order $q \in Z_n^*$. Bob sends p, q, g and v to Alice.

Commitment phase:

A COMMITMENT SCHEME BASED on DISCRETE LOGARITHM

Alice wants to commit herself to an $m \in \{0, \ldots, q-1\}$.

Scheme setting:

Bob randomly chooses primes p and q such that

$$q | (p-1).$$

Bob chooses random generators $g \neq 1 \neq v$ of the subgroup G of order $q \in Z_n^*$. Bob sends p, q, g and v to Alice.

Commitment phase:

To commit to an $m \in \{0, \ldots, q-1\}$, Alice chooses a random $r \in Z_q$, and sends $c = g^r v^m$ to Bob.

A COMMITMENT SCHEME BASED on DISCRETE LOGARITHM

Alice wants to commit herself to an $m \in \{0, \ldots, q-1\}$.

Scheme setting:

Bob randomly chooses primes p and q such that

$$q|(p-1).$$

Bob chooses random generators $g \neq 1 \neq v$ of the subgroup G of order $q \in Z_n^*$. Bob sends p, q, g and v to Alice.

Commitment phase:

To commit to an $m \in \{0, \ldots, q-1\}$, Alice chooses a random $r \in Z_q$, and sends $c = g^r v^m$ to Bob.

Opening phase:

Alice sends r and m to Bob who then verifies whether c$= g^r v^m$.

COMMENTS

- If Alice, committed to an m, could open her commitment as $\bar{m} \neq m$, using some $\bar{r}$, then $g^r v^m = g^{\bar{r}} v^{\bar{m}}$ and therefore
$$\lg_g v = (r - \bar{r})(\bar{m} - m)^{-1}.$$
Hence, Alice could compute $lg_g v$ of a randomly chosen element $v \in G$, what contradicts the assumption that computation of discrete logarithms in G is infeasible.

COMMENTS

- If Alice, committed to an m, could open her commitment as $\bar{m} \neq m$, using some $\bar{r}$, then $g^r v^m = g^{\bar{r}} v^{\bar{m}}$ and therefore
$$\lg_g v = (r - \bar{r})(\bar{m} - m)^{-1}.$$
Hence, Alice could compute $lg_g v$ of a randomly chosen element $v \in G$, what contradicts the assumption that computation of discrete logarithms in G is infeasible.
- Since g and v are generators of G, then g^r is a uniformly chosen random element in G, perfectly hiding v^m and m in $g^r v^m$, as in the encryption with ONE-TIME PAD cryptosystem.

BIT COMMITMENT using ENCRYPTIONS

Commit phase:

1. Bob generates a random string r and sends it to Alice
2. Alice commit herself to a bit b using a key k through an encryption

$$E_k(rb)$$

   and sends it to Bob.

BIT COMMITMENT using ENCRYPTIONS

Commit phase:

1. Bob generates a random string r and sends it to Alice
2. Alice commit herself to a bit b using a key k through an encryption

$$E_k(rb)$$

and sends it to Bob.

Opening phase:

1. Alice sends the key k to Bob.
2. Bob decrypts the message to learn b and to verify r.

BIT COMMITMENT using ENCRYPTIONS

Commit phase:

1. Bob generates a random string r and sends it to Alice
2. Alice commit herself to a bit b using a key k through an encryption

$$E_k(rb)$$

   and sends it to Bob.

Opening phase:

1. Alice sends the key k to Bob.
2. Bob decrypts the message to learn b and to verify r.

Comment: without Bob's random string r Alice could find a different key l such that $e_k(b) = e_l(\neg b)$.

Let com(r, m) $= g^r v^m$ denote commitment to m in the commitment scheme based on discrete logarithm. If $r_1, r_2, m_1, m_2 \in Z_n$, then $com(r_1, m_1) \times com(r_2, m_2) = com(r_1 + r_2, m_1 + m_2)$. Commitment schemes with such a property are called homomorphic commitment schemes.

COMMITMENTS and ELECTRONIC VOTING

Let com(r, m) $= g^r v^m$ denote commitment to m in the commitment scheme based on discrete logarithm. If $r_1, r_2, m_1, m_2 \in Z_n$, then $com(r_1, m_1) \times com(r_2, m_2) = com(r_1 + r_2, m_1 + m_2)$. Commitment schemes with such a property are called homomorphic commitment schemes. Homomorphic schemes can be used to cast yes-no votes of n voters $V_1, \ldots, V_n$, by the trusted authority TA for whom e_T and d_T are ElGamal encryption and decryption algorithms.

Let com(r, m) $= g^r v^m$ denote commitment to m in the commitment scheme based on discrete logarithm. If $r_1, r_2, m_1, m_2 \in Z_n$, then $com(r_1, m_1) \times com(r_2, m_2) = com(r_1 + r_2, m_1 + m_2)$.
Commitment schemes with such a property are called homomorphic commitment schemes.
Homomorphic schemes can be used to cast yes-no votes of n voters $V_1, \ldots, V_n$, by the trusted authority TA for whom e_T and d_T are ElGamal encryption and decryption algorithms.
This works as follows: Each voter V_i chooses his vote $m_i \in \{0, 1\}$, a random $r_I \in \{0, \ldots, q-1\}$ and computes his voting commitment $c_I = com(r_i, m_i)$. Then V_i makes c_i public and sends $e_T(g^{r_i})$ to TA and TA computes

$$d_T\left(\prod_{i=1}^{n} e_T(g^{r_i})\right) = \prod_{i=1}^{n} g^{r_i} = g^r,$$

where $r = \sum_{i=1}^{n} r_i$, and makes public g^r.

COMMITMENTS and ELECTRONIC VOTING

Let com(r, m) $= g^r v^m$ denote commitment to m in the commitment scheme based on discrete logarithm. If $r_1, r_2, m_1, m_2 \in Z_n$, then $com(r_1, m_1) \times com(r_2, m_2) = com(r_1 + r_2, m_1 + m_2)$.
Commitment schemes with such a property are called homomorphic commitment schemes.
Homomorphic schemes can be used to cast yes-no votes of n voters $V_1, \ldots, V_n$, by the trusted authority TA for whom e_T and d_T are ElGamal encryption and decryption algorithms.
This works as follows: Each voter V_i chooses his vote $m_i \in \{0, 1\}$, a random $r_I \in \{0, \ldots, q-1\}$ and computes his voting commitment $c_I = com(r_i, m_i)$. Then V_i makes c_i public and sends $e_T(g^{r_i})$ to TA and TA computes

$$d_T\left(\prod_{i=1}^{n} e_T(g^{r_i})\right) = \prod_{i=1}^{n} g^{r_i} = g^r,$$

where $r = \sum_{i=1}^{n} r_i$, and makes public g^r.

Now, anybody can compute the result s of voting from publicly known c_i and g^r since

$$v^s = \frac{\prod_{i=1}^{n} c_i}{g^r},$$

with $s = \sum_{i=1}^{n} m_i$.

s can now be derived from v^s by computing $v^1, v^2, v^3, \ldots$ and comparing with v^s if the number of voters is not too large.

TRUST in CRYPTOGRAPHIC PROTOCOLS - deliberations

In any interaction between people, there is a certain level of risk, trust, and expected behaviour, that is implicit in the interchanges.

In any interaction between people, there is a certain level of risk, trust, and expected behaviour, that is implicit in the interchanges.

People may behave properly for a variety of reasons: fear from prosecution, desire to act in unethical manner due to social influences, and so on.

TRUST in CRYPTOGRAPHIC PROTOCOLS - deliberations

In any interaction between people, there is a certain level of risk, trust, and expected behaviour, that is implicit in the interchanges.

People may behave properly for a variety of reasons: fear from prosecution, desire to act in unethical manner due to social influences, and so on.

However, in cryptographic protocols trust has to be kept to the lowest possible level.

TRUST in CRYPTOGRAPHIC PROTOCOLS - deliberations

In any interaction between people, there is a certain level of risk, trust, and expected behaviour, that is implicit in the interchanges.

People may behave properly for a variety of reasons: fear from prosecution, desire to act in unethical manner due to social influences, and so on.

However, in cryptographic protocols trust has to be kept to the lowest possible level.

In any cryptographic protocol, if there is an absence of a mechanism for verifying, say authenticity, one must assume, as default, that other participants can be dishonest (if for no other reason than for self-preservation).

OBLIVIOUS TRANSFER (OT) PROBLEM

Story: Alice knows a secret and wants to send secret to Bob in such a way that he gets secret with probability $\frac{1}{2}$, and he knows whether he got secret, but Alice has no idea whether he received secret. (Or Alice has several secrets and Bob wants to buy one of them but he does not want Alice to know which one he bought.)

OBLIVIOUS TRANSFER (OT) PROBLEM

Story: Alice knows a secret and wants to send secret to Bob in such a way that he gets secret with probability $\frac{1}{2}$, and he knows whether he got secret, but Alice has no idea whether he received secret. (Or Alice has several secrets and Bob wants to buy one of them but he does not want Alice to know which one he bought.)

Oblivious transfer problem: Design a protocol for sending a message from Alice to Bob in such a way that Bob receives the message with probability $\frac{1}{2}$ and "garbage" with the probability $\frac{1}{2}$. Moreover, Bob knows whether he got the message or garbage, but Alice has no idea which one he got.

OBLIVIOUS TRANSFER PROTOCOL - continuation

Oblivious transfer problem: Design a protocol for sending a message from Alice to Bob in such a way that Bob receives the message with probability $\frac{1}{2}$ and "garbage" with the probability $\frac{1}{2}$. Moreover, Bob knows whether he got the message or garbage, but Alice has no idea which one he got.

An Oblivious transfer protocol:

1. Alice chooses two large primes p and q and sends $n = pq$ to Bob.

Oblivious transfer problem: Design a protocol for sending a message from Alice to Bob in such a way that Bob receives the message with probability $\frac{1}{2}$ and "garbage" with the probability $\frac{1}{2}$. Moreover, Bob knows whether he got the message or garbage, but Alice has no idea which one he got.

An Oblivious transfer protocol:

1. Alice chooses two large primes p and q and sends $n = pq$ to Bob.
2. Bob chooses a random number x and sends $y = x^2 \mod n$ to Alice.

OBLIVIOUS TRANSFER PROTOCOL - continuation

Oblivious transfer problem: Design a protocol for sending a message from Alice to Bob in such a way that Bob receives the message with probability $\frac{1}{2}$ and "garbage" with the probability $\frac{1}{2}$. Moreover, Bob knows whether he got the message or garbage, but Alice has no idea which one he got.

An Oblivious transfer protocol:

1. Alice chooses two large primes p and q and sends $n = pq$ to Bob.
2. Bob chooses a random number x and sends $y = x^2 \mod n$ to Alice.
3. Alice computes four square roots $\pm x_1, \pm x_2$ of y (mod n) and sends one of them to Bob. (She can do it, but has no idea which of them is x.)

Oblivious transfer problem: Design a protocol for sending a message from Alice to Bob in such a way that Bob receives the message with probability $\frac{1}{2}$ and "garbage" with the probability $\frac{1}{2}$. Moreover, Bob knows whether he got the message or garbage, but Alice has no idea which one he got.

An Oblivious transfer protocol:

1. Alice chooses two large primes p and q and sends $n = pq$ to Bob.
2. Bob chooses a random number x and sends $y = x^2 \mod n$ to Alice.
3. Alice computes four square roots $\pm x_1, \pm x_2$ of y (mod n) and sends one of them to Bob. (She can do it, but has no idea which of them is x.)
4. Bob checks whether the number he got is congruent to x. If yes, he has received no new information. Otherwise, Bob has two different square roots modulo n and can factor n. Alice has no way of knowing whether this is the case.

OBLIVIOUS TRANSFER PROTOCOL - continuation

Oblivious transfer problem: Design a protocol for sending a message from Alice to Bob in such a way that Bob receives the message with probability $\frac{1}{2}$ and "garbage" with the probability $\frac{1}{2}$. Moreover, Bob knows whether he got the message or garbage, but Alice has no idea which one he got.

An Oblivious transfer protocol:

1. Alice chooses two large primes p and q and sends $n = pq$ to Bob.
2. Bob chooses a random number x and sends $y = x^2 \mod n$ to Alice.
3. Alice computes four square roots $\pm x_1, \pm x_2$ of y (mod n) and sends one of them to Bob. (She can do it, but has no idea which of them is x.)
4. Bob checks whether the number he got is congruent to x. If yes, he has received no new information. Otherwise, Bob has two different square roots modulo n and can factor n. Alice has no way of knowing whether this is the case.

1-OUT-OF-2 OBLIVIOUS TRANSFER PROBLEM

The 1-out-of-2 oblivious transfer problem: Alice sends two messages to Bob in such a way that Bob can choose which of the messages he receives (but he cannot choose both), but Alice cannot learn Bob's decision.

1-OUT-OF-2 OBLIVIOUS TRANSFER PROBLEM

The 1-out-of-2 oblivious transfer problem: Alice sends two messages to Bob in such a way that Bob can choose which of the messages he receives (but he cannot choose both), but Alice cannot learn Bob's decision.

A generalization of 1-out-of-2 oblivious transfer problem is two-party oblivious circuit evaluation problem:

1-OUT-OF-2 OBLIVIOUS TRANSFER PROBLEM

The 1-out-of-2 oblivious transfer problem: Alice sends two messages to Bob in such a way that Bob can choose which of the messages he receives (but he cannot choose both), but Alice cannot learn Bob's decision.

A generalization of 1-out-of-2 oblivious transfer problem is two-party oblivious circuit evaluation problem:

Alice has a secret i and Bob has a secret j and they both know some function f.

At the end of protocol the following conditions should hold:

1. Bob knows the value f(i,j), but he does not learn anything about i.
2. Alice learns nothing about j and nothing about f(i,j).

1-OUT-OF-2 OBLIVIOUS TRANSFER PROBLEM

The 1-out-of-2 oblivious transfer problem: Alice sends two messages to Bob in such a way that Bob can choose which of the messages he receives (but he cannot choose both), but Alice cannot learn Bob's decision.

A generalization of 1-out-of-2 oblivious transfer problem is two-party oblivious circuit evaluation problem:

Alice has a secret i and Bob has a secret j and they both know some function f.

At the end of protocol the following conditions should hold:

1. Bob knows the value f(i,j), but he does not learn anything about i.
2. Alice learns nothing about j and nothing about f(i,j).

Note: The 1-out-of-2 oblivious transfer problem is the instance of the oblivious circuit evaluation problem for $i = (b_0, b_1), f(i,j) = b_j$.

1-out-2 OBLIVIOUS TRANSFER BOX

1-out-of-two oblivious transfer can be imagined as a box with three inputs and one output.

1-out-2 OBLIVIOUS TRANSFER BOX

1-out-of-two oblivious transfer can be imagined as a box with three inputs and one output.

INPUTS: Alice inputs: x_0 and x_1;
.... Bob inputs a bit c

1-out-2 OBLIVIOUS TRANSFER BOX

1-out-of-two oblivious transfer can be imagined as a box with three inputs and one output.

INPUTS: Alice inputs: x_0 and x_1;
.... Bob inputs a bit c

OUTPUT: Bob gets as the output: x_c

- Alice generates two key pairs for a PKC P and sends both her public keys p_1, p_2 to Bob.

AN IMPLEMENTATION of OBLIVIOUS TRANSFER PROTOCOLS

- Alice generates two key pairs for a PKC P and sends both her public keys p_1, p_2 to Bob.
- Bob chooses a to-be random secret key k for a SKC S, encrypts it by one of Alice's public keys, p_1 or p_2 and sends the outcome to Alice.

AN IMPLEMENTATION of OBLIVIOUS TRANSFER PROTOCOLS

- Alice generates two key pairs for a PKC P and sends both her public keys p_1, p_2 to Bob.
- Bob chooses a to-be random secret key k for a SKC S, encrypts it by one of Alice's public keys, p_1 or p_2 and sends the outcome to Alice.
- Alice uses her two secret keys to decrypt the message she received. One of the outcomes is garbage g, another one is k, but she does not know which one is k.

- Alice generates two key pairs for a PKC P and sends both her public keys p_1, p_2 to Bob.
- Bob chooses a to-be random secret key k for a SKC S, encrypts it by one of Alice's public keys, p_1 or p_2 and sends the outcome to Alice.
- Alice uses her two secret keys to decrypt the message she received. One of the outcomes is garbage g, another one is k, but she does not know which one is k.
- Alice encrypts her two secret messages, one with k, another with g and sends them to Bob.

- Alice generates two key pairs for a PKC P and sends both her public keys p_1, p_2 to Bob.
- Bob chooses a to-be random secret key k for a SKC S, encrypts it by one of Alice's public keys, p_1 or p_2 and sends the outcome to Alice.
- Alice uses her two secret keys to decrypt the message she received. One of the outcomes is garbage g, another one is k, but she does not know which one is k.
- Alice encrypts her two secret messages, one with k, another with g and sends them to Bob.
- Bob uses S with k to decrypt both messages he got and one of the attempts is successful. Alice has no idea which one.

HISTORY and POWER of OBLIVIOUS TRANSFER PROTOCOLS

- C. Crépeau (1988) showed that both versions of oblivious transfer are equivalent – a protocol for each version can be realized using any protocol for the other version, using a cryptographic reduction

HISTORY and POWER of OBLIVIOUS TRANSFER PROTOCOLS

- C. Crépeau (1988) showed that both versions of oblivious transfer are equivalent – a protocol for each version can be realized using any protocol for the other version, using a cryptographic reduction
- Original definition of the oblivious transfer is due to J. Halpern and M. O. Rabin (1983); 1-out-of-2 oblivious transfer suggested S. Even, O. Goldreich and A. Lempel in 1985.

HISTORY and POWER of OBLIVIOUS TRANSFER PROTOCOLS

- C. Crépeau (1988) showed that both versions of oblivious transfer are equivalent – a protocol for each version can be realized using any protocol for the other version, using a cryptographic reduction
- Original definition of the oblivious transfer is due to J. Halpern and M. O. Rabin (1983); 1-out-of-2 oblivious transfer suggested S. Even, O. Goldreich and A. Lempel in 1985.
- J. Kilian (1988) showed that oblivious transfers are very powerful protocols that allow secure computation of the value $f(x, y)$ of any binary function f, where x is a secret value known only by Alice, and y is a secret value known only by Bob, in such a way that it holds:
  - Both, Alice and Bob, learn $f(x, y)$
  - Alice learns about y only as much as she can learn from x and $f(x, y)$
  - Bob learns about x only as much as he can learn from y and $f(x, y)$

BIT COMMITMENT from 1-out-2 oblivious transfer

Using 1-out-of-2 oblivious transfer box (OT-box) one can design a bit commitment scheme:

BIT COMMITMENT from 1-out-2 oblivious transfer

Using 1-out-of-2 oblivious transfer box (OT-box) one can design a bit commitment scheme:

COMMITMENT PHASE:

1. Alice selects a random bit r and her commitment bit b;
2. Alice inputs $x_0 = r$ and $x_1 = r \oplus b$ into the OT-box.
3. Alice sends a message to Bob telling him it is his turn.
4. Bob selects a random bit c, inputs c into the OT-box and records the output x_c.

BIT COMMITMENT from 1-out-2 oblivious transfer

Using 1-out-of-2 oblivious transfer box (OT-box) one can design a bit commitment scheme:

COMMITMENT PHASE:

1. Alice selects a random bit r and her commitment bit b;
2. Alice inputs $x_0 = r$ and $x_1 = r \oplus b$ into the OT-box.
3. Alice sends a message to Bob telling him it is his turn.
4. Bob selects a random bit c, inputs c into the OT-box and records the output x_c.

OPENING PHASE:

1. Alice sends r and b to Bob.
2. Bob checks to see if $x_c = r \oplus (bc)$

MENTAL POKER PLAYING by PHONE by Alice and Bob

Basic requirements (for playing poker with 52 cards):

- Initial hands (sets of 5 cards) of both players are equally likely.
- The initial hands of Alice and Bob are disjoint.
- Both players always know their own hands but not that of the opponent.
- Each player can detect eventual cheating of the other player.

MENTAL POKER PLAYING by PHONE by Alice and Bob

Basic requirements (for playing poker with 52 cards):

- Initial hands (sets of 5 cards) of both players are equally likely.
- The initial hands of Alice and Bob are disjoint.
- Both players always know their own hands but not that of the opponent.
- Each player can detect eventual cheating of the other player.

A commutative cryptosystem is used with all functions kept secret.

Players agree on numbers $w_1, \ldots, w_{52}$ as the names of 52 cards.

MENTAL POKER PLAYING by PHONE by Alice and Bob

Basic requirements (for playing poker with 52 cards):

- Initial hands (sets of 5 cards) of both players are equally likely.
- The initial hands of Alice and Bob are disjoint.
- Both players always know their own hands but not that of the opponent.
- Each player can detect eventual cheating of the other player.

A commutative cryptosystem is used with all functions kept secret.

Players agree on numbers $w_1, \dots, w_{52}$ as the names of 52 cards.

Protocol:

1. Bob encrypts cards with e_B, and tells $e_B(w_1), \dots, e_B(w_{52})$, in a randomly chosen order, to Alice.

MENTAL POKER PLAYING by PHONE by Alice and Bob

Basic requirements (for playing poker with 52 cards):

- Initial hands (sets of 5 cards) of both players are equally likely.
- The initial hands of Alice and Bob are disjoint.
- Both players always know their own hands but not that of the opponent.
- Each player can detect eventual cheating of the other player.

A commutative cryptosystem is used with all functions kept secret.

Players agree on numbers $w_1, \ldots, w_{52}$ as the names of 52 cards.

Protocol:

1. Bob encrypts cards with e_B, and tells $e_B(w_1), \ldots, e_B(w_{52})$, in a randomly chosen order, to Alice.
2. Alice chooses five of the items $e_B(w_i)$ as Bob's hand and tells them Bob.

MENTAL POKER PLAYING by PHONE by Alice and Bob

Basic requirements (for playing poker with 52 cards):

- Initial hands (sets of 5 cards) of both players are equally likely.
- The initial hands of Alice and Bob are disjoint.
- Both players always know their own hands but not that of the opponent.
- Each player can detect eventual cheating of the other player.

A commutative cryptosystem is used with all functions kept secret.

Players agree on numbers $w_1, \dots, w_{52}$ as the names of 52 cards.

Protocol:

1. Bob encrypts cards with e_B, and tells $e_B(w_1), \dots, e_B(w_{52})$, in a randomly chosen order, to Alice.
2. Alice chooses five of the items $e_B(w_i)$ as Bob's hand and tells them Bob.
3. Alice chooses another five of $e_B(w_i)$, encrypts them with e_A and sends them to Bob.

MENTAL POKER PLAYING by PHONE by Alice and Bob

Basic requirements (for playing poker with 52 cards):

- Initial hands (sets of 5 cards) of both players are equally likely.
- The initial hands of Alice and Bob are disjoint.
- Both players always know their own hands but not that of the opponent.
- Each player can detect eventual cheating of the other player.

A commutative cryptosystem is used with all functions kept secret.

Players agree on numbers $w_1, \ldots, w_{52}$ as the names of 52 cards.

Protocol:

1. Bob encrypts cards with e_B, and tells $e_B(w_1), \ldots, e_B(w_{52})$, in a randomly chosen order, to Alice.
2. Alice chooses five of the items $e_B(w_i)$ as Bob's hand and tells them Bob.
3. Alice chooses another five of $e_B(w_i)$, encrypts them with e_A and sends them to Bob.
4. Bob applies d_B to all five values $e_A(e_B(w_i))$ he got from Alice and sends $e_A(w_i)$ to Alice as Alice's hand. At this point both players have their hands and poker can start.

MENTAL POKER PLAYING by PHONE by Alice and Bob

Basic requirements (for playing poker with 52 cards):

- Initial hands (sets of 5 cards) of both players are equally likely.
- The initial hands of Alice and Bob are disjoint.
- Both players always know their own hands but not that of the opponent.
- Each player can detect eventual cheating of the other player.

A commutative cryptosystem is used with all functions kept secret.

Players agree on numbers $w_1, \ldots, w_{52}$ as the names of 52 cards.

Protocol:

1. Bob encrypts cards with e_B, and tells $e_B(w_1), \ldots, e_B(w_{52})$, in a randomly chosen order, to Alice.
2. Alice chooses five of the items $e_B(w_i)$ as Bob's hand and tells them Bob.
3. Alice chooses another five of $e_B(w_i)$, encrypts them with e_A and sends them to Bob.
4. Bob applies d_B to all five values $e_A(e_B(w_i))$ he got from Alice and sends $e_A(w_i)$ to Alice as Alice's hand. At this point both players have their hands and poker can start.

Remark: The cryptosystems that are used cannot be public-key in the normal sense. Otherwise Alice could compute $e_B(w_i)$ and deal with the cards accordingly – a good hand for B but slightly better for herself.

MENTAL POKER by PHONE with THREE PLAYERS

1. Alice encrypts 52 cards $w_1, \ldots, w_{52}$ with e_A and sends encryptions, in a random order, to Bob.

MENTAL POKER by PHONE with THREE PLAYERS

1. Alice encrypts 52 cards $w_1, \ldots, w_{52}$ with e_A and sends encryptions, in a random order, to Bob.
2. Bob, who cannot decode the encryptions, chooses 5 of them, randomly. He encrypts them with e_B, and sends $e_B(e_A(w_i))$ to Alice and the remaining 47 encryptions $e_A(w_i)$ to Carol.

MENTAL POKER by PHONE with THREE PLAYERS

1. Alice encrypts 52 cards $w_1, \ldots, w_{52}$ with e_A and sends encryptions, in a random order, to Bob.
2. Bob, who cannot decode the encryptions, chooses 5 of them, randomly. He encrypts them with e_B, and sends $e_B(e_A(w_i))$ to Alice and the remaining 47 encryptions $e_A(w_i)$ to Carol.
3. Carol, who cannot decode any of the encryptions, chooses five of them randomly, encrypts them also with her key and sends Alice $e_C(e_A(w_i))$.

MENTAL POKER by PHONE with THREE PLAYERS

1. Alice encrypts 52 cards $w_1, \ldots, w_{52}$ with e_A and sends encryptions, in a random order, to Bob.
2. Bob, who cannot decode the encryptions, chooses 5 of them, randomly. He encrypts them with e_B, and sends $e_B(e_A(w_i))$ to Alice and the remaining 47 encryptions $e_A(w_i)$ to Carol.
3. Carol, who cannot decode any of the encryptions, chooses five of them randomly, encrypts them also with her key and sends Alice $e_C(e_A(w_i))$.
4. Alice, who cannot read encrypted messages from Bob and Carol, decrypt them with her key and sends back to the senders,

$$\text{five } d_A(e_B(e_A(w_i))) = e_B(w_i) \text{ to Bob,}$$
$$\text{five } d_A(e_C(e_A(w_i))) = e_C(w_i) \text{ to Carol.}$$

MENTAL POKER by PHONE with THREE PLAYERS

1. Alice encrypts 52 cards $w_1, \ldots, w_{52}$ with e_A and sends encryptions, in a random order, to Bob.
2. Bob, who cannot decode the encryptions, chooses 5 of them, randomly. He encrypts them with e_B, and sends $e_B(e_A(w_i))$ to Alice and the remaining 47 encryptions $e_A(w_i)$ to Carol.
3. Carol, who cannot decode any of the encryptions, chooses five of them randomly, encrypts them also with her key and sends Alice $e_C(e_A(w_i))$.
4. Alice, who cannot read encrypted messages from Bob and Carol, decrypt them with her key and sends back to the senders,

$$\text{five } d_A(e_B(e_A(w_i))) = e_B(w_i) \text{ to Bob,}$$
$$\text{five } d_A(e_C(e_A(w_i))) = e_C(w_i) \text{ to Carol.}$$

5. Bob and Carol decrypt encryptions they got to learn their hands.
6. Carol chooses randomly 5 other messages $e_A(w_i)$ from the remaining 42 and sends them to Alice.

MENTAL POKER by PHONE with THREE PLAYERS

1. Alice encrypts 52 cards $w_1, \ldots, w_{52}$ with e_A and sends encryptions, in a random order, to Bob.
2. Bob, who cannot decode the encryptions, chooses 5 of them, randomly. He encrypts them with e_B, and sends $e_B(e_A(w_i))$ to Alice and the remaining 47 encryptions $e_A(w_i)$ to Carol.
3. Carol, who cannot decode any of the encryptions, chooses five of them randomly, encrypts them also with her key and sends Alice $e_C(e_A(w_i))$.
4. Alice, who cannot read encrypted messages from Bob and Carol, decrypt them with her key and sends back to the senders,

$$\text{five } d_A(e_B(e_A(w_i))) = e_B(w_i) \text{ to Bob,}$$
$$\text{five } d_A(e_C(e_A(w_i))) = e_C(w_i) \text{ to Carol.}$$

5. Bob and Carol decrypt encryptions they got to learn their hands.
6. Carol chooses randomly 5 other messages $e_A(w_i)$ from the remaining 42 and sends them to Alice.
7. Alice decrypt messages to learn her hand.

Additional cards can be dealt with in a similar manner. If either Bob or Carol wants a card, they take an encrypted message $e_A(w_i)$ and go through the protocol with Alice. If Alice wants a card, whoever currently has the deck sends her a card.

ZERO-KNOWLEDGE PROOF PROTOCOLS

To the most important primitives for cryptographic protocols, and at the same time very counter intuitive primitives, belong so-called zero-knowledge (proof) protocols.

ZERO-KNOWLEDGE PROOF PROTOCOLS

To the most important primitives for cryptographic protocols, and at the same time very counter intuitive primitives, belong so-called zero-knowledge (proof) protocols.

Very informally, a zero-knowledge proof protocol allows one party, usually called PROVER, to convince another party, called VERIFIER, that PROVER knows some fact (a secret, a proof of a theorem,...) without revealing to the VERIFIER **ANY** information about his knowledge (secret, proof,...).

In the rest of this chapter we present and illustrate very basic ideas of zero-knowledge proof protocols and their importance for cryptography.

ZERO-KNOWLEDGE PROOF PROTOCOLS

To the most important primitives for cryptographic protocols, and at the same time very counter intuitive primitives, belong so-called zero-knowledge (proof) protocols.

Very informally, a zero-knowledge proof protocol allows one party, usually called PROVER, to convince another party, called VERIFIER, that PROVER knows some fact (a secret, a proof of a theorem,...) without revealing to the VERIFIER **ANY** information about his knowledge (secret, proof,...).

In the rest of this chapter we present and illustrate very basic ideas of zero-knowledge proof protocols and their importance for cryptography.

Zero-knowledge proof protocols are a special type of so-called interactive proof systems.

ZERO-KNOWLEDGE PROOF PROTOCOLS

To the most important primitives for cryptographic protocols, and at the same time very counter intuitive primitives, belong so-called zero-knowledge (proof) protocols.

Very informally, a zero-knowledge proof protocol allows one party, usually called PROVER, to convince another party, called VERIFIER, that PROVER knows some fact (a secret, a proof of a theorem,...) without revealing to the VERIFIER **ANY** information about his knowledge (secret, proof,...).

In the rest of this chapter we present and illustrate very basic ideas of zero-knowledge proof protocols and their importance for cryptography.

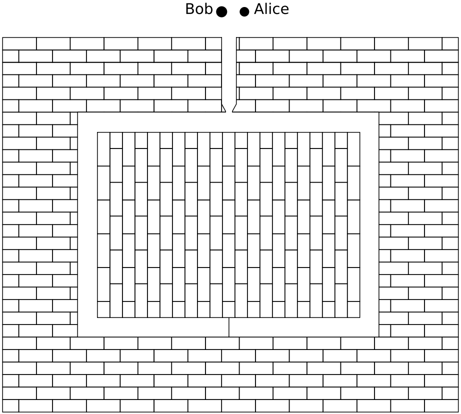
Zero-knowledge proof protocols are a special type of so-called interactive proof systems.

By a theorem we understand in the following a claim that a specific object has a specific property. For example, that a specific graph is 3-colorable.

AN ILLUSTRATIVE EXAMPLE

(A cave with a door opening on a secret word)

Alice knows a secret word opening the door in cave. How can she convince Bob about it without revealing this secret word?



ZERO-KNOWLEDGE PROOFS

Informally speaking, an interactive proof systems has the property of being zero-knowledge if the Verifier, that interacts with the honest Prover of the system, learns nothing from their interaction beyond the validity of the statement being proved.

ZERO-KNOWLEDGE PROOFS

Informally speaking, an interactive proof systems has the property of being zero-knowledge if the Verifier, that interacts with the honest Prover of the system, learns nothing from their interaction beyond the validity of the statement being proved.

There are several variants of zero-knowledge protocols that differ in the specific way the notion of learning nothing is formalized.

In each variant it is viewed that a particular Verifier learns nothing if there exists a polynomial time simulator whose output is indistinguishable from the output of the Verifier after interacting with the Prover on any possible instance of the problem.

ZERO-KNOWLEDGE PROOFS

Informally speaking, an interactive proof systems has the property of being zero-knowledge if the Verifier, that interacts with the honest Prover of the system, learns nothing from their interaction beyond the validity of the statement being proved.

There are several variants of zero-knowledge protocols that differ in the specific way the notion of learning nothing is formalized.

In each variant it is viewed that a particular Verifier learns nothing if there exists a polynomial time simulator whose output is indistinguishable from the output of the Verifier after interacting with the Prover on any possible instance of the problem.

The different variants of zero-knowledge proof systems concern the strength of this distinguishability. In particular, perfect or statistical zero-knowledge refer to the situation where the simulator's output and the Verifier's output are indistinguishable in an information theoretic sense.

ZERO-KNOWLEDGE PROOFS

Informally speaking, an interactive proof systems has the property of being zero-knowledge if the Verifier, that interacts with the honest Prover of the system, learns nothing from their interaction beyond the validity of the statement being proved.

There are several variants of zero-knowledge protocols that differ in the specific way the notion of learning nothing is formalized.

In each variant it is viewed that a particular Verifier learns nothing if there exists a polynomial time simulator whose output is indistinguishable from the output of the Verifier after interacting with the Prover on any possible instance of the problem.

The different variants of zero-knowledge proof systems concern the strength of this distinguishability. In particular, perfect or statistical zero-knowledge refer to the situation where the simulator's output and the Verifier's output are indistinguishable in an information theoretic sense.

Computational zero-knowledge refer to the case there is no polynomial time distinguishability.

INTERACTIVE PROOF PROTOCOLS

In an interactive proof system there are two parties

- An (all powerful) Prover, often called Peggy (a randomized algorithm that uses a private random number generator);
- A (little (polynomially) powerful) Verifier, often called Vic (a polynomial time randomized algorithm that uses a private random number generator).

INTERACTIVE PROOF PROTOCOLS

In an interactive proof system there are two parties

- An (all powerful) Prover, often called Peggy (a randomized algorithm that uses a private random number generator);
- A (little (polynomially) powerful) Verifier, often called Vic (a polynomial time randomized algorithm that uses a private random number generator).

Prover knows some secret, or a knowledge, or a fact about a specific object, and wishes to convince Vic, through a communication with him, that he has the above knowledge.

INTERACTIVE PROOF PROTOCOLS

In an interactive proof system there are two parties

- An (all powerful) Prover, often called Peggy (a randomized algorithm that uses a private random number generator);
- A (little (polynomially) powerful) Verifier, often called Vic (a polynomial time randomized algorithm that uses a private random number generator).

Prover knows some secret, or a knowledge, or a fact about a specific object, and wishes to convince Vic, through a communication with him, that he has the above knowledge.

For example, both Prover and Verifier posses an input x and Prover wants to convince Verifier that x has a certain Property and that Prover knows how to prove that.

INTERACTIVE PROOF PROTOCOLS

In an interactive proof system there are two parties

- An (all powerful) Prover, often called Peggy (a randomized algorithm that uses a private random number generator);
- A (little (polynomially) powerful) Verifier, often called Vic (a polynomial time randomized algorithm that uses a private random number generator).

Prover knows some secret, or a knowledge, or a fact about a specific object, and wishes to convince Vic, through a communication with him, that he has the above knowledge.

For example, both Prover and Verifier posses an input x and Prover wants to convince Verifier that x has a certain Property and that Prover knows how to prove that.

The interactive proof system consists of several rounds. In each round Prover and Verifier alternatively do the following.

1. Receive a message from the other party.
2. Perform a (private) computation.
3. Send a message to the other party.

Communication starts usually by a challenge of Verifier and a response of Prover.

At the end, Verifier either accepts or rejects Prover's attempts to convince Verifier.

EXAMPLE – GRAPH NON-ISOMORPHISM

A simple interactive proof protocol exists for a computationally very hard graph non-isomorphism problem.

Input: Two graphs G_1 and G_2, with the set of nodes $\{1, \ldots, n\}$

EXAMPLE – GRAPH NON-ISOMORPHISM

A simple interactive proof protocol exists for a computationally very hard graph non-isomorphism problem.

Input: Two graphs G_1 and G_2, with the set of nodes $\{1, \ldots, n\}$

Protocol: Repeat n times the following steps:

1. Vic chooses randomly an integer $i \in \{1, 2\}$ and a permutation π of $\{1, \ldots, n\}$. Vic then computes the image H of G_i under permutation π and sends H to Peggy.
2. Peggy determines the value j such that G_J is isomorphic to H, and sends j to Vic.
3. Vic checks to see if $\mathrm{i} = \mathrm{j}$.

Vic accepts Peggy's proof if $\mathrm{i} = \mathrm{j}$ in each of n rounds.

EXAMPLE – GRAPH NON-ISOMORPHISM

A simple interactive proof protocol exists for a computationally very hard graph non-isomorphism problem.

Input: Two graphs G_1 and G_2, with the set of nodes $\{1, \ldots, n\}$

Protocol: Repeat n times the following steps:

1. Vic chooses randomly an integer $i \in \{1, 2\}$ and a permutation π of $\{1, \ldots, n\}$. Vic then computes the image H of G_i under permutation π and sends H to Peggy.
2. Peggy determines the value j such that G_J is isomorphic to H, and sends j to Vic.
3. Vic checks to see if $\mathrm{i} = \mathrm{j}$.

Vic accepts Peggy's proof if $\mathrm{i} = \mathrm{j}$ in each of n rounds.

Completeness: If G_1 is not isomorphic to G_2, then probability that Vic accepts is clearly 1 because Peggy will have no problem answer correctly.

EXAMPLE – GRAPH NON-ISOMORPHISM

A simple interactive proof protocol exists for a computationally very hard graph non-isomorphism problem.

Input: Two graphs G_1 and G_2, with the set of nodes $\{1, \ldots, n\}$

Protocol: Repeat n times the following steps:

1. Vic chooses randomly an integer $i \in \{1, 2\}$ and a permutation π of $\{1, \ldots, n\}$. Vic then computes the image H of G_i under permutation π and sends H to Peggy.
2. Peggy determines the value j such that G_J is isomorphic to H, and sends j to Vic.
3. Vic checks to see if $\mathrm{i} = \mathrm{j}$.

Vic accepts Peggy's proof if $\mathrm{i} = \mathrm{j}$ in each of n rounds.

Completeness: If G_1 is not isomorphic to G_2, then probability that Vic accepts is clearly 1 because Peggy will have no problem answer correctly.

Soundness: If G_1 is isomorphic to G_2, then Peggy can deceive Vic if and only if she correctly guesses n times those i's Vic chooses randomly. Probability that this happens is 2^{-n}.

Observe that Vic's computations can be performed in polynomial time (with respect to the size of graphs).

INTERACTIVE PROOF SYSTEMS

An interactive proof protocol is said to be an interactive proof system for a secret/knowledge or a decision problem Π if the following properties are satisfied provided that Prover and Verifier posses an input x (or Prover has secret knowledge) and Prover wants to convince Verifier that x has certain properties and that Prover knows how to prove that (or that Prover knows the secret).

INTERACTIVE PROOF SYSTEMS

An interactive proof protocol is said to be an interactive proof system for a secret/knowledge or a decision problem Π if the following properties are satisfied provided that Prover and Verifier posses an input x (or Prover has secret knowledge) and Prover wants to convince Verifier that x has certain properties and that Prover knows how to prove that (or that Prover knows the secret).

(Knowledge) Completeness: If x is a yes-instance of Π, or Peggy knows the secret, then Vic always accepts Peggy's "proof" for sure.

INTERACTIVE PROOF SYSTEMS

An interactive proof protocol is said to be an interactive proof system for a secret/knowledge or a decision problem Π if the following properties are satisfied provided that Prover and Verifier posses an input x (or Prover has secret knowledge) and Prover wants to convince Verifier that x has certain properties and that Prover knows how to prove that (or that Prover knows the secret).

(Knowledge) Completeness: If x is a yes-instance of Π, or Peggy knows the secret, then Vic always accepts Peggy's "proof" for sure.

(Knowledge) Soundness: If x is a no-instance of Π, or Peggy does not know the secret, then Vic accepts Peggy's "proof" only with very small probability.

INTERACTIVE PROOF SYSTEMS

An interactive proof protocol is said to be an interactive proof system for a secret/knowledge or a decision problem Π if the following properties are satisfied provided that Prover and Verifier posses an input x (or Prover has secret knowledge) and Prover wants to convince Verifier that x has certain properties and that Prover knows how to prove that (or that Prover knows the secret).

(Knowledge) Completeness: If x is a yes-instance of Π, or Peggy knows the secret, then Vic always accepts Peggy's "proof" for sure.

(Knowledge) Soundness: If x is a no-instance of Π, or Peggy does not know the secret, then Vic accepts Peggy's "proof" only with very small probability.

CHEATING

- If the Prover and the Verifier of an interactive proof system fully follow the protocol they are called honest Prover and honest Verifier.
- A Prover who does not know secret or proof and tries to convince the Verifier is called cheating Prover.
- A Verifier who does not follow the behaviour specified in the protocol is called a cheating Verifier.

ZERO-KNOWLEDGE PROOF PROTOCOLS INFORMATION VERY INFORMALLY

Very informally An interactive "proof protocol" at which a Prover tries to convince a Verifier about the truth of a statement, or about possession of a knowledge, is called "zero-knowledge" protocol if the Verifier does not learn from communication anything more except that the statement is true or that Prover has knowledge (secret) she claims to have.

ZERO-KNOWLEDGE PROOF PROTOCOLS INFORMATION VERY INFORMALLY

Very informally An interactive "proof protocol" at which a Prover tries to convince a Verifier about the truth of a statement, or about possession of a knowledge, is called "zero-knowledge" protocol if the Verifier does not learn from communication anything more except that the statement is true or that Prover has knowledge (secret) she claims to have. Example The proof $n = 670592745 = 12345 \times 54321$ is not a zero-knowledge proof that n is not a prime.

ZERO-KNOWLEDGE PROOF PROTOCOLS INFORMATION MORE FORMALLY

huge **Informally** A zero-knowledge proof is an interactive proof protocol that provides **highly convincing evidence** that a statement is true or that Prover has certain knowledge (of a secret) and that Prover knows a (standard) proof of it while providing **not a single bit of information** about the proof (knowledge or secret). (In particular, Verifier who got convinced about the correctness of a statement cannot convince the third person about that.)

ZERO-KNOWLEDGE PROOF PROTOCOLS INFORMATION MORE FORMALLY

huge **Informally** A zero-knowledge proof is an interactive proof protocol that provides **highly convincing evidence** that a statement is true or that Prover has certain knowledge (of a secret) and that Prover knows a (standard) proof of it while providing **not a single bit of information** about the proof (knowledge or secret). (In particular, Verifier who got convinced about the correctness of a statement cannot convince the third person about that.)
More formally A zero-knowledge proof of a theorem T is an interactive two party protocol, in which Prover is able to convince Verifier who follows the same protocol, by the overwhelming statistical evidence, that T is true, if T is indeed true, but no Prover is able to convince Verifier that T is true, if this is not so. In addition, during interactions, Prover does not reveal to Verifier any other information, except whether T is true or not. Consequently, whatever Verifier can do after he gets convinced, he can do just believing that T is true.
Similar arguments hold for the case Prover possesses a secret.

AGE DIFFERENCE FINDING PROTOCOL

Alice and Bob want to find out who of them is older without disclosing any other information about their age.

AGE DIFFERENCE FINDING PROTOCOL

Alice and Bob want to find out who of them is older without disclosing any other information about their age.

The following protocol is based on a public-key cryptosystem, in which it is assumed that neither Bob nor Alice are older than 100 years.

AGE DIFFERENCE FINDING PROTOCOL

Alice and Bob want to find out who of them is older without disclosing any other information about their age.

The following protocol is based on a public-key cryptosystem, in which it is assumed that neither Bob nor Alice are older than 100 years.

Protocol Let age of Bob be j; and age of Alice be i.

1. Bob chooses a random $x \in \{1, \ldots, 100\}$, computes $k = e_A(x)$ and sends to Alice s = k - j.

AGE DIFFERENCE FINDING PROTOCOL

Alice and Bob want to find out who of them is older without disclosing any other information about their age.

The following protocol is based on a public-key cryptosystem, in which it is assumed that neither Bob nor Alice are older than 100 years.

Protocol Let age of Bob be j; and age of Alice be i.

1. Bob chooses a random $x \in \{1, \ldots, 100\}$, computes $k = e_A(x)$ and sends to Alice s = k - j.
2. Alice first computes the numbers $y_u = d_A(s + u); 1 \leq u \leq 100$, then chooses a large random prime p and computes numbers
$$z_u = y_u \mod p, \qquad 1 \leq u \leq 100 \qquad (*)$$
and verifies that for all $u \neq v$
$$|z_u - z_v| \geq 2 \text{ and } z_u \neq 0 \qquad (**)$$
(If this is not the case, Alice choose a new p, repeats computations in (*) and checks (**) again.)

AGE DIFFERENCE FINDING PROTOCOL

Alice and Bob want to find out who of them is older without disclosing any other information about their age.

The following protocol is based on a public-key cryptosystem, in which it is assumed that neither Bob nor Alice are older than 100 years.

Protocol Let age of Bob be j; and age of Alice be i.

1. Bob chooses a random $x \in \{1, \ldots, 100\}$, computes $k = e_A(x)$ and sends to Alice s = k - j.
2. Alice first computes the numbers $y_u = d_A(s + u); 1 \leq u \leq 100$, then chooses a large random prime p and computes numbers

   $$z_u = y_u \mod p, \qquad 1 \leq u \leq 100 \qquad\qquad (*)$$

   and verifies that for all $u \neq v$

   $$|z_u - z_v| \geq 2 \text{ and } z_u \neq 0 \qquad\qquad (**)$$

   (If this is not the case, Alice choose a new p, repeats computations in (*) and checks (**) again.)

   Finally, Alice sends Bob the following sequence (order is important).

   $$z_1, \ldots, z_i, z_{i+1} + 1, \ldots, z_{100} + 1, p$$
   $$\text{as } z'_1, \ldots, z'_i, z'_{i+1}, \ldots, z'_{100}, p$$

AGE DIFFERENCE FINDING PROTOCOL

Alice and Bob want to find out who of them is older without disclosing any other information about their age.

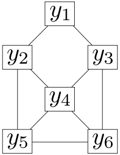
The following protocol is based on a public-key cryptosystem, in which it is assumed that neither Bob nor Alice are older than 100 years.

Protocol Let age of Bob be j; and age of Alice be i.

1. Bob chooses a random $x \in \{1, \ldots, 100\}$, computes $k = e_A(x)$ and sends to Alice s = k - j.
2. Alice first computes the numbers $y_u = d_A(s + u); 1 \leq u \leq 100$, then chooses a large random prime p and computes numbers
$$z_u = y_u \mod p, \qquad 1 \leq u \leq 100 \qquad\qquad (*)$$
and verifies that for all $u \neq v$
$$|z_u - z_v| \geq 2 \text{ and } z_u \neq 0 \qquad\qquad (**)$$
(If this is not the case, Alice choose a new p, repeats computations in (*) and checks (**) again.)
Finally, Alice sends Bob the following sequence (order is important).
$$z_1, \ldots, z_i, z_{i+1} + 1, \ldots, z_{100} + 1, p$$
$$\text{as } z'_1, \ldots, z'_i, z'_{i+1}, \ldots, z'_{100}, p$$
3. Bob checks whether j-th number in the above sequence is congruent to x modulo p. If yes, Bob knows that $i \geq j$, otherwise $i < j$.
$$i \geq j \Rightarrow z'_J = z_J \equiv y_J = d_A(k) \equiv x \pmod p$$
$$i < j \Rightarrow z'_J = z_J + 1 \not\equiv y_J = d_A(k) \equiv x \pmod p$$

3-COLORABILITY of GRAPHS

With the following protocol Peggy can convince Vic that a particular graph G, known to both of them, is 3-colorable and that Peggy knows such a coloring, without revealing to Vic any information how such coloring looks.



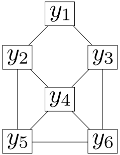
(a)

1 red	e_1	$e_1(red) = y_1$
2 green	e_2	$e_2(green) = y_2$
3 blue	e_3	$e_3(blue) = y_3$
4 red	e_4	$e_4(red) = y_4$
5 blue	e_5	$e_5(blue) = y_5$
6 green	e_6	$e_6(green) = y_6$

(b)

3-COLORABILITY of GRAPHS

With the following protocol Peggy can convince Vic that a particular graph G, known to both of them, is 3-colorable and that Peggy knows such a coloring, without revealing to Vic any information how such coloring looks.



(a)

1	red	e_1	$e_1(red) = y_1$
2	green	e_2	$e_2(green) = y_2$
3	blue	e_3	$e_3(blue) = y_3$
4	red	e_4	$e_4(red) = y_4$
5	blue	e_5	$e_5(blue) = y_5$
6	green	e_6	$e_6(green) = y_6$

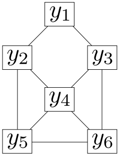
(b)

Protocol: Peggy colors the graph $G = (V, E)$ with colors (red, blue, green) and she performs with Vic $|E|^2$- times the following interactions, where $v_1, \ldots, v_n$ are nodes of V.

1. Peggy chooses a random permutation of colors, recolors G, and encrypts, for i = 1,2,. . . ,n, the color c_i of node v_i by an encryption procedure e_i – for each i different.

3-COLORABILITY of GRAPHS

With the following protocol Peggy can convince Vic that a particular graph G, known to both of them, is 3-colorable and that Peggy knows such a coloring, without revealing to Vic any information how such coloring looks.



(a)

1 red	e_1	$e_1(red) = y_1$
2 green	e_2	$e_2(green) = y_2$
3 blue	e_3	$e_3(blue) = y_3$
4 red	e_4	$e_4(red) = y_4$
5 blue	e_5	$e_5(blue) = y_5$
6 green	e_6	$e_6(green) = y_6$

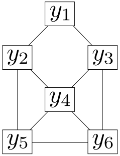
(b)

Protocol: Peggy colors the graph $G = (V, E)$ with colors (red, blue, green) and she performs with Vic $|E|^2$- times the following interactions, where $v_1, \ldots, v_n$ are nodes of V.

1. Peggy chooses a random permutation of colors, recolors G, and encrypts, for i = 1,2,. . . ,n, the color c_i of node v_i by an encryption procedure e_i – for each i different. Peggy then removes colors from nodes, labels the i-th node of G with cryptotext $y_i = e_i(c_i)$, and designs Table (b).

3-COLORABILITY of GRAPHS

With the following protocol Peggy can convince Vic that a particular graph G, known to both of them, is 3-colorable and that Peggy knows such a coloring, without revealing to Vic any information how such coloring looks.



(a)

1 red	e_1	$e_1(red) = y_1$
2 green	e_2	$e_2(green) = y_2$
3 blue	e_3	$e_3(blue) = y_3$
4 red	e_4	$e_4(red) = y_4$
5 blue	e_5	$e_5(blue) = y_5$
6 green	e_6	$e_6(green) = y_6$

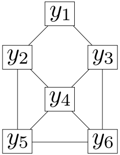
(b)

Protocol: Peggy colors the graph $G = (V, E)$ with colors (red, blue, green) and she performs with Vic $|E|^2$- times the following interactions, where $v_1, \ldots, v_n$ are nodes of V.

1. Peggy chooses a random permutation of colors, recolors G, and encrypts, for i = 1,2,. . . ,n, the color c_i of node v_i by an encryption procedure e_i – for each i different. Peggy then removes colors from nodes, labels the i-th node of G with cryptotext $y_i = e_i(c_i)$, and designs Table (b).
   Peggy finally shows Vic the graph with nodes labeled by cryptotexts.

3-COLORABILITY of GRAPHS

With the following protocol Peggy can convince Vic that a particular graph G, known to both of them, is 3-colorable and that Peggy knows such a coloring, without revealing to Vic any information how such coloring looks.



(a)

1 red	e_1	$e_1(red) = y_1$
2 green	e_2	$e_2(green) = y_2$
3 blue	e_3	$e_3(blue) = y_3$
4 red	e_4	$e_4(red) = y_4$
5 blue	e_5	$e_5(blue) = y_5$
6 green	e_6	$e_6(green) = y_6$

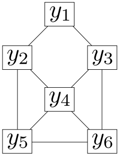
(b)

Protocol: Peggy colors the graph $G = (V, E)$ with colors (red, blue, green) and she performs with Vic $|E|^2$- times the following interactions, where $v_1, \ldots, v_n$ are nodes of V.

1. Peggy chooses a random permutation of colors, recolors G, and encrypts, for i = 1,2,. . . ,n, the color c_i of node v_i by an encryption procedure e_i – for each i different. Peggy then removes colors from nodes, labels the i-th node of G with cryptotext $y_i = e_i(c_i)$, and designs Table (b).
   Peggy finally shows Vic the graph with nodes labeled by cryptotexts.
2. Vic chooses an edge and asks Peggy to show him coloring of the corresponding nodes.

3-COLORABILITY of GRAPHS

With the following protocol Peggy can convince Vic that a particular graph G, known to both of them, is 3-colorable and that Peggy knows such a coloring, without revealing to Vic any information how such coloring looks.



(a)

1	red	e_1	$e_1(red) = y_1$
2	green	e_2	$e_2(green) = y_2$
3	blue	e_3	$e_3(blue) = y_3$
4	red	e_4	$e_4(red) = y_4$
5	blue	e_5	$e_5(blue) = y_5$
6	green	e_6	$e_6(green) = y_6$

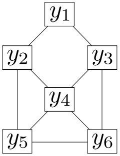
(b)

Protocol: Peggy colors the graph $G = (V, E)$ with colors (red, blue, green) and she performs with Vic $|E|^2$- times the following interactions, where $v_1, \ldots, v_n$ are nodes of V.

1. Peggy chooses a random permutation of colors, recolors G, and encrypts, for i = 1,2,. . . ,n, the color c_i of node v_i by an encryption procedure e_i – for each i different. Peggy then removes colors from nodes, labels the i-th node of G with cryptotext $y_i = e_i(c_i)$, and designs Table (b).
   Peggy finally shows Vic the graph with nodes labeled by cryptotexts.
2. Vic chooses an edge and asks Peggy to show him coloring of the corresponding nodes.
3. Peggy shows Vic entries of the table corresponding to the nodes of the chosen edge.

3-COLORABILITY of GRAPHS

With the following protocol Peggy can convince Vic that a particular graph G, known to both of them, is 3-colorable and that Peggy knows such a coloring, without revealing to Vic any information how such coloring looks.



(a)

1 red	e_1	$e_1(red) = y_1$
2 green	e_2	$e_2(green) = y_2$
3 blue	e_3	$e_3(blue) = y_3$
4 red	e_4	$e_4(red) = y_4$
5 blue	e_5	$e_5(blue) = y_5$
6 green	e_6	$e_6(green) = y_6$

(b)

Protocol: Peggy colors the graph G = (V, E) with colors (red, blue, green) and she performs with Vic $|E|^2$- times the following interactions, where $v_1, \ldots, v_n$ are nodes of V.

1. Peggy chooses a random permutation of colors, recolors G, and encrypts, for i = 1,2,. . . ,n, the color c_i of node v_i by an encryption procedure e_i – for each i different. Peggy then removes colors from nodes, labels the i-th node of G with cryptotext $y_i = e_i(c_i)$, and designs Table (b).
   Peggy finally shows Vic the graph with nodes labeled by cryptotexts.
2. Vic chooses an edge and asks Peggy to show him coloring of the corresponding nodes.
3. Peggy shows Vic entries of the table corresponding to the nodes of the chosen edge.
4. Vic performs desired encryptions to verify that nodes really have colors as shown.

APPLICATIONS of ZERO-KNOWLEDGE PROOFS in CRYPTOGRAPHIC PROTOCOLS

The fact that for a big class of statements there are zero-knowledge proofs can be used to design secure cryptographic protocols. (All languages in NP have zero-knowledge proofs.)

APPLICATIONS of ZERO-KNOWLEDGE PROOFS in CRYPTOGRAPHIC PROTOCOLS

The fact that for a big class of statements there are zero-knowledge proofs can be used to design secure cryptographic protocols. (All languages in NP have zero-knowledge proofs.)

A cryptographic protocol can be seen as a set of interactive programs to be executed by non-trusting parties.

APPLICATIONS of ZERO-KNOWLEDGE PROOFS in CRYPTOGRAPHIC PROTOCOLS

The fact that for a big class of statements there are zero-knowledge proofs can be used to design secure cryptographic protocols. (All languages in NP have zero-knowledge proofs.)

A cryptographic protocol can be seen as a set of interactive programs to be executed by non-trusting parties.

Each party keeps secret her local input.

APPLICATIONS of ZERO-KNOWLEDGE PROOFS in CRYPTOGRAPHIC PROTOCOLS

The fact that for a big class of statements there are zero-knowledge proofs can be used to design secure cryptographic protocols. (All languages in NP have zero-knowledge proofs.)

A cryptographic protocol can be seen as a set of interactive programs to be executed by non-trusting parties.

Each party keeps secret her local input.

The protocol specifies the actions parties should take, depending on their local secrets and previous messages exchanged.

APPLICATIONS of ZERO-KNOWLEDGE PROOFS in CRYPTOGRAPHIC PROTOCOLS

The fact that for a big class of statements there are zero-knowledge proofs can be used to design secure cryptographic protocols. (All languages in NP have zero-knowledge proofs.)

A cryptographic protocol can be seen as a set of interactive programs to be executed by non-trusting parties.

Each party keeps secret her local input.

The protocol specifies the actions parties should take, depending on their local secrets and previous messages exchanged.

The main problem in this setting is how can a party verify that the other parties have really followed the protocol?

APPLICATIONS of ZERO-KNOWLEDGE PROOFS in CRYPTOGRAPHIC PROTOCOLS

The fact that for a big class of statements there are zero-knowledge proofs can be used to design secure cryptographic protocols. (All languages in NP have zero-knowledge proofs.)

A cryptographic protocol can be seen as a set of interactive programs to be executed by non-trusting parties.

Each party keeps secret her local input.

The protocol specifies the actions parties should take, depending on their local secrets and previous messages exchanged.

The main problem in this setting is how can a party verify that the other parties have really followed the protocol?

The way out: a party A can convince a party B that the transmitted message was completed according to the protocol without revealing its secrets.

APPLICATIONS of ZERO-KNOWLEDGE PROOFS in CRYPTOGRAPHIC PROTOCOLS

The fact that for a big class of statements there are zero-knowledge proofs can be used to design secure cryptographic protocols. (All languages in NP have zero-knowledge proofs.)

A cryptographic protocol can be seen as a set of interactive programs to be executed by non-trusting parties.

Each party keeps secret her local input.

The protocol specifies the actions parties should take, depending on their local secrets and previous messages exchanged.

The main problem in this setting is how can a party verify that the other parties have really followed the protocol?

The way out: a party A can convince a party B that the transmitted message was completed according to the protocol without revealing its secrets.

An idea how to design a reliable protocol

1. Design a protocol under the assumption that all parties follow the protocol.

APPLICATIONS of ZERO-KNOWLEDGE PROOFS in CRYPTOGRAPHIC PROTOCOLS

The fact that for a big class of statements there are zero-knowledge proofs can be used to design secure cryptographic protocols. (All languages in NP have zero-knowledge proofs.)

A cryptographic protocol can be seen as a set of interactive programs to be executed by non-trusting parties.

Each party keeps secret her local input.

The protocol specifies the actions parties should take, depending on their local secrets and previous messages exchanged.

The main problem in this setting is how can a party verify that the other parties have really followed the protocol?

The way out: a party A can convince a party B that the transmitted message was completed according to the protocol without revealing its secrets.

An idea how to design a reliable protocol

1. Design a protocol under the assumption that all parties follow the protocol.
2. Transform protocol, using known methods how to make zero-knowledge proofs out of normal ones, into a protocol in which communication is based on zero-knowledge proofs, and which preserves both correctness and privacy and works even if some parties display an adversary behavior.

ZERO-KNOWLEDGE PROOF for QUADRATIC RESIDUA

Input: An integer $n = pq$, where p, q are primes and $x \in QR(n)$.

ZERO-KNOWLEDGE PROOF for QUADRATIC RESIDUA

Input: An integer $n = pq$, where p, q are primes and $x \in QR(n)$.

Protocol: Repeat $\lg n$ times the following steps:

ZERO-KNOWLEDGE PROOF for QUADRATIC RESIDUA

Input: An integer $n = pq$, where p, q are primes and $x \in QR(n)$.

Protocol: Repeat $\lg n$ times the following steps:

1. Peggy chooses a random $v \in Z_n^*$ and sends to Vic

$$y = v^2 \mod n.$$

ZERO-KNOWLEDGE PROOF for QUADRATIC RESIDUA

Input: An integer $n = pq$, where p, q are primes and $x \in QR(n)$.

Protocol: Repeat $\lg n$ times the following steps:

1. Peggy chooses a random $v \in Z_n^*$ and sends to Vic
$$y = v^2 \mod n.$$
2. Vic sends to Peggy a random $i \in \{0, 1\}$.

ZERO-KNOWLEDGE PROOF for QUADRATIC RESIDUA

Input: An integer $n = pq$, where p, q are primes and $x \in QR(n)$.

Protocol: Repeat $\lg n$ times the following steps:

1. Peggy chooses a random $v \in Z_n^*$ and sends to Vic
$$y = v^2 \mod n.$$
2. Vic sends to Peggy a random $i \in \{0, 1\}$.
3. Peggy computes a square root u of x and sends to Vic
$$z = u^i v \mod n.$$

ZERO-KNOWLEDGE PROOF for QUADRATIC RESIDUA

Input: An integer $n = pq$, where p, q are primes and $x \in QR(n)$.

Protocol: Repeat $\lg n$ times the following steps:

1. Peggy chooses a random $v \in Z_n^*$ and sends to Vic
$$y = v^2 \mod n.$$
2. Vic sends to Peggy a random $i \in \{0, 1\}$.
3. Peggy computes a square root u of x and sends to Vic
$$z = u^i v \mod n.$$
4. Vic checks whether
$$z^2 \equiv x^i y \mod n.$$

Vic accepts Peggy's proof that x is QR if he succeeds in point 4 in each of $\lg n$ rounds.

ZERO-KNOWLEDGE PROOF for QUADRATIC RESIDUA

Input: An integer $n = pq$, where p, q are primes and $x \in QR(n)$.

Protocol: Repeat $\lg n$ times the following steps:

1. Peggy chooses a random $v \in Z_n^*$ and sends to Vic
$$y = v^2 \mod n.$$
2. Vic sends to Peggy a random $i \in \{0, 1\}$.
3. Peggy computes a square root u of x and sends to Vic
$$z = u^i v \mod n.$$
4. Vic checks whether
$$z^2 \equiv x^i y \mod n.$$

Vic accepts Peggy's proof that x is QR if he succeeds in point 4 in each of $\lg n$ rounds.

Completeness is straightforward:

Soundness If x is not a quadratic residue, then Peggy can answer only one of two possible challenges (only if $i = 0$), because in such a case y is a quadratic residue if and only if xy is not a quadratic residue.This means that Peggy will be caught in any given round of the protocol with probability $\frac{1}{2}$.

The overall probability that prover deceives Vic is therefore $2^{-\lg n} = \frac{1}{n}$.

ZERO-KNOWLEDGE PROOF for GRAPH ISOMORPHISM

Input: Given are two graphs G_1 and G_2 with the set of nodes $\{1, \ldots, n\}$.

Repeat the following steps n times:

ZERO-KNOWLEDGE PROOF for GRAPH ISOMORPHISM

Input: Given are two graphs G_1 and G_2 with the set of nodes $\{1, \ldots, n\}$.

Repeat the following steps n times:

1. Peggy chooses a random permutation π of $\{1, \ldots, n\}$ and computes H to be the image of G_1 under the permutation π, and sends H to Vic.

ZERO-KNOWLEDGE PROOF for GRAPH ISOMORPHISM

Input: Given are two graphs G_1 and G_2 with the set of nodes $\{1, \ldots, n\}$.

Repeat the following steps n times:

1. Peggy chooses a random permutation π of $\{1, \ldots, n\}$ and computes H to be the image of G_1 under the permutation π, and sends H to Vic.
2. Vic chooses randomly $i \in \{1, 2\}$ and sends it to Peggy. {This way Vic asks for isomorphism between H and G_i.}

ZERO-KNOWLEDGE PROOF for GRAPH ISOMORPHISM

Input: Given are two graphs G_1 and G_2 with the set of nodes $\{1, \ldots, n\}$.

Repeat the following steps n times:

1. Peggy chooses a random permutation π of $\{1, \ldots, n\}$ and computes H to be the image of G_1 under the permutation π, and sends H to Vic.
2. Vic chooses randomly $i \in \{1, 2\}$ and sends it to Peggy. {This way Vic asks for isomorphism between H and G_i.}
3. Peggy creates a permutation ρ of $\{1, \ldots, n\}$ such that ρ specifies isomorphism between H and G_i and Peggy sends ρ to Vic.

   {If $i = 1$ Peggy takes $\rho = \pi$; if $i = 2$ Peggy takes $\rho = \sigma o \pi$, where σ is a fixed isomorphic mapping of nodes of G_2 to G_1.}

ZERO-KNOWLEDGE PROOF for GRAPH ISOMORPHISM

Input: Given are two graphs G_1 and G_2 with the set of nodes $\{1, \ldots, n\}$.

Repeat the following steps n times:

1. Peggy chooses a random permutation π of $\{1, \ldots, n\}$ and computes H to be the image of G_1 under the permutation π, and sends H to Vic.
2. Vic chooses randomly $i \in \{1, 2\}$ and sends it to Peggy. {This way Vic asks for isomorphism between H and G_i.}
3. Peggy creates a permutation ρ of $\{1, \ldots, n\}$ such that ρ specifies isomorphism between H and G_i and Peggy sends ρ to Vic.
   {If $\mathrm{i} = 1$ Peggy takes $\rho = \pi$; if $\mathrm{i} = 2$ Peggy takes $\rho = \sigma o \pi$, where σ is a fixed isomorphic mapping of nodes of G_2 to G_1.}
4. Vic checks whether H provides the isomorphism between G_i and H.
   Vic accepts Peggy's "proof" if H is the image of G_i in each of the n rounds.

ZERO-KNOWLEDGE PROOF for GRAPH ISOMORPHISM

Input: Given are two graphs G_1 and G_2 with the set of nodes $\{1, \ldots, n\}$.

Repeat the following steps n times:

1. Peggy chooses a random permutation π of $\{1, \ldots, n\}$ and computes H to be the image of G_1 under the permutation π, and sends H to Vic.
2. Vic chooses randomly $i \in \{1, 2\}$ and sends it to Peggy. $\{$This way Vic asks for isomorphism between H and G_i.$\}$
3. Peggy creates a permutation ρ of $\{1, \ldots, n\}$ such that ρ specifies isomorphism between H and G_i and Peggy sends ρ to Vic.
   $\{$If $\mathrm{i} = 1$ Peggy takes $\rho = \pi$; if $\mathrm{i} = 2$ Peggy takes $\rho = \sigma o \pi$, where σ is a fixed isomorphic mapping of nodes of G_2 to G_1.$\}$
4. Vic checks whether H provides the isomorphism between G_i and H.
   Vic accepts Peggy's "proof" if H is the image of G_i in each of the n rounds.

Completeness. It is obvious that if G_1 and G_2 are isomorphic then Vic accepts with probability 1.

ZERO-KNOWLEDGE PROOF for GRAPH ISOMORPHISM

Input: Given are two graphs G_1 and G_2 with the set of nodes $\{1, \ldots, n\}$.
Repeat the following steps n times:

1. Peggy chooses a random permutation π of $\{1, \ldots, n\}$ and computes H to be the image of G_1 under the permutation π, and sends H to Vic.
2. Vic chooses randomly $i \in \{1, 2\}$ and sends it to Peggy. {This way Vic asks for isomorphism between H and G_i.}
3. Peggy creates a permutation ρ of $\{1, \ldots, n\}$ such that ρ specifies isomorphism between H and G_i and Peggy sends ρ to Vic.
   {If $i = 1$ Peggy takes $\rho = \pi$; if $i = 2$ Peggy takes $\rho = \sigma o \pi$, where σ is a fixed isomorphic mapping of nodes of G_2 to G_1.}
4. Vic checks whether H provides the isomorphism between G_i and H.
   Vic accepts Peggy's "proof" if H is the image of G_i in each of the n rounds.

Completeness. It is obvious that if G_1 and G_2 are isomorphic then Vic accepts with probability 1.

Soundness: If graphs G_1 and G_2 are not isomorphic, then Peggy can deceive Vic only if she is able to guess in each round the i Vic chooses and then sends as H the graph G_i. However, the probability that this happens is 2^{-n}.

Observe that Vic can perform all computations in polynomial time. However, why is this proof a zero-knowledge proof?

WHY is the last "PROOF" a "ZERO-KNOWLEDGE PROOF"?

Because Vic gets convinced, by the overwhelming statistical evidence, that graphs G_1 and G_2 are isomorphic, but he does not get any information ("knowledge") that would help him to create isomorphism between G_1 and G_2.

WHY is the last "PROOF" a "ZERO-KNOWLEDGE PROOF"?

Because Vic gets convinced, by the overwhelming statistical evidence, that graphs G_1 and G_2 are isomorphic, but he does not get any information ("knowledge") that would help him to create isomorphism between G_1 and G_2.

In each round of the proof Vic see isomorphism between H (a random isomorphic copy of G_1) and G_1 or G_2, (but not between both of them)!

WHY is the last "PROOF" a "ZERO-KNOWLEDGE PROOF"?

Because Vic gets convinced, by the overwhelming statistical evidence, that graphs G_1 and G_2 are isomorphic, but he does not get any information ("knowledge") that would help him to create isomorphism between G_1 and G_2.

In each round of the proof Vic see isomorphism between H (a random isomorphic copy of G_1) and G_1 or G_2, (but not between both of them)!

However, Vic can create such random copies H of the graphs by himself and therefore it seems very unlikely that this can help Vic to find an isomorphism between G_1 and G_2.

WHY is the last "PROOF" a "ZERO-KNOWLEDGE PROOF"?

Because Vic gets convinced, by the overwhelming statistical evidence, that graphs G_1 and G_2 are isomorphic, but he does not get any information ("knowledge") that would help him to create isomorphism between G_1 and G_2.

In each round of the proof Vic see isomorphism between H (a random isomorphic copy of G_1) and G_1 or G_2, (but not between both of them)!

However, Vic can create such random copies H of the graphs by himself and therefore it seems very unlikely that this can help Vic to find an isomorphism between G_1 and G_2.

Information that Vic can receive during the protocol, called transcript, contains:

- The graphs G_1 and G_2.
- All messages i transmitted during communications by Peggy and Vic.
- Random numbers r used by Peggy and Vic to generate their outputs.

WHY is the last "PROOF" a "ZERO-KNOWLEDGE PROOF"?

Because Vic gets convinced, by the overwhelming statistical evidence, that graphs G_1 and G_2 are isomorphic, but he does not get any information ("knowledge") that would help him to create isomorphism between G_1 and G_2.

In each round of the proof Vic see isomorphism between H (a random isomorphic copy of G_1) and G_1 or G_2, (but not between both of them)!

However, Vic can create such random copies H of the graphs by himself and therefore it seems very unlikely that this can help Vic to find an isomorphism between G_1 and G_2.

Information that Vic can receive during the protocol, called transcript, contains:

- The graphs G_1 and G_2.
- All messages i transmitted during communications by Peggy and Vic.
- Random numbers r used by Peggy and Vic to generate their outputs.

Transcript has therefore the form

$$T = ((G_1, G_2); (H_1, i_1, r_1), \ldots, (H_n, i_n, r_n)).$$

The essential point, which is the basis for the formal definition of zero-knowledge proof, is that Vic can forge transcript, without participating in the interactive proof, that look like "real transcripts", if graphs are isomorphic, by means of the following forging algorithm called simulator.

SIMULATOR

A simulator for the previous graph isomorphism protocol.

- $T = (G_1, G_2)$,
- **for** $\mathsf{j} = 1$ **to** n **do**

SIMULATOR

A simulator for the previous graph isomorphism protocol.

- $T = (G_1, G_2)$,
- **for** j $= 1$ **to** n **do**
  - Chose randomly $i_j \in \{1, 2\}$.
  - Chose ρ_j to be a random permutation of $\{1, \ldots, n\}$.
  - Compute H_j to be the image of G_{i_j} under ρ_j;
  - Concatenate (H_j, i_j, ρ_j) at the end of T.

CONSEQUENCES and FORMAL DEFINITION

The fact that a simulator can forge transcripts has several important consequences.

- Anything Vic can compute using the information obtained from the transcript can be computed using only a forged transcript and therefore participation in such a communication does not increase Vic capability to perform any computation.
- Participation in such a proof does not allow Vic to prove isomorphism of G_1 and G_2.
- Vic cannot convince someone else that G_1 and G_2 are isomorphic by showing the transcript because it is indistinguishable from a forged one.

CONSEQUENCES and FORMAL DEFINITION

The fact that a simulator can forge transcripts has several important consequences.

- Anything Vic can compute using the information obtained from the transcript can be computed using only a forged transcript and therefore participation in such a communication does not increase Vic capability to perform any computation.
- Participation in such a proof does not allow Vic to prove isomorphism of G_1 and G_2.
- Vic cannot convince someone else that G_1 and G_2 are isomorphic by showing the transcript because it is indistinguishable from a forged one.

Formal definition of what this means that a forged transcript "looks like" a real one:

Definition Suppose that we have an interactive proof system for a decision problem Π and a polynomial time simulator S.

Denote by $\Gamma(x)$ the set of all possible transcripts that could be produced during the interactive proof communication for a yes-instance x.

Denote F(x) the set of all possible forged transcripts produced by the simulator S.

For any transcript $T \in \Gamma(x)$, let $p_\Gamma(T)$ denote the probability that T is the transcript produced during the interactive proof. Similarly, for $T \in F(x)$, let $p_F(T)$ denote the probability that T is the transcript produced by S.

If $\Gamma(x) = F(x)$ and, for any $T \in \Gamma(x)$, $p_\Gamma(T) = p_F(T)$, then we say that the interactive proof system is a zero-knowledge proof system.

WHY is the last "PROOF" a "ZERO-KNOWLEDGE PROOF"?

Because Vic gets convinced, by the overwhelming statistical evidence, that graphs G_1 and G_2 are isomorphic, but he does not get any information ("knowledge") that would help him to create isomorphism between G_1 and G_2.

WHY is the last "PROOF" a "ZERO-KNOWLEDGE PROOF"?

Because Vic gets convinced, by the overwhelming statistical evidence, that graphs G_1 and G_2 are isomorphic, but he does not get any information ("knowledge") that would help him to create isomorphism between G_1 and G_2.

In each round of the proof Vic see isomorphism between H (a random isomorphic copy of G_1) and G_1 or G_2, (but not between both of them)!

WHY is the last "PROOF" a "ZERO-KNOWLEDGE PROOF"?

Because Vic gets convinced, by the overwhelming statistical evidence, that graphs G_1 and G_2 are isomorphic, but he does not get any information ("knowledge") that would help him to create isomorphism between G_1 and G_2.

In each round of the proof Vic see isomorphism between H (a random isomorphic copy of G_1) and G_1 or G_2, (but not between both of them)!

However, Vic can create such random copies H of the graphs by himself and therefore it seems very unlikely that this can help Vic to find an isomorphism between G_1 and G_2.

WHY is the last "PROOF" a "ZERO-KNOWLEDGE PROOF"?

Because Vic gets convinced, by the overwhelming statistical evidence, that graphs G_1 and G_2 are isomorphic, but he does not get any information ("knowledge") that would help him to create isomorphism between G_1 and G_2.

In each round of the proof Vic see isomorphism between H (a random isomorphic copy of G_1) and G_1 or G_2, (but not between both of them)!

However, Vic can create such random copies H of the graphs by himself and therefore it seems very unlikely that this can help Vic to find an isomorphism between G_1 and G_2.

Information that Vic can receive during the protocol, called transcript, contains:

- The graphs G_1 and G_2.
- All messages i transmitted during communications by Peggy and Vic.
- Random numbers r used by Peggy and Vic to generate their outputs.

WHY is the last "PROOF" a "ZERO-KNOWLEDGE PROOF"?

Because Vic gets convinced, by the overwhelming statistical evidence, that graphs G_1 and G_2 are isomorphic, but he does not get any information ("knowledge") that would help him to create isomorphism between G_1 and G_2.
In each round of the proof Vic see isomorphism between H (a random isomorphic copy of G_1) and G_1 or G_2, (but not between both of them)!

However, Vic can create such random copies H of the graphs by himself and therefore it seems very unlikely that this can help Vic to find an isomorphism between G_1 and G_2.
Information that Vic can receive during the protocol, called transcript, contains:

- The graphs G_1 and G_2.
- All messages i transmitted during communications by Peggy and Vic.
- Random numbers r used by Peggy and Vic to generate their outputs.

Transcript has therefore the form

$$T = ((G_1, G_2); (H_1, i_1, r_1), \dots, (H_n, i_n, r_n)).$$

The essential point, which is the basis for the formal definition of zero-knowledge proof, is that Vic can forge transcript, without participating in the interactive proof, that look like "real transcripts", if graphs are isomorphic, by means of the following forging algorithm called simulator.

SIMULATOR

A simulator for the previous graph isomorphism protocol.

- $T = (G_1, G_2)$,
- **for** j $=$ 1 **to** n **do**
  - Chose randomly $i_j \in \{1, 2\}$.
  - Chose ρ_j to be a random permutation of $\{1, \ldots, n\}$.
  - Compute H_j to be the image of G_{i_j} under ρ_j;
  - Concatenate (H_j, i_j, ρ_j) at the end of T.
- If, in an interactive proof system, the probability distributions specified by the protocols with Vic and with simulator are computationally indistinguishable in polynomial time , then we speak about computationally zero-knowledge proof system.

Part XI

Steganography and Watermarking

DIGITAL STEGANOGRAPHY and DIGITAL WATERMARJIN

A very important property of (digital) information is that it is, in principle, very easy to produce and distribute unlimited number of its copies.

DIGITAL STEGANOGRAPHY and DIGITAL WATERMARJIN

A very important property of (digital) information is that it is, in principle, very easy to produce and distribute unlimited number of its copies.

This might undermine the music, film, book and software industries and therefore it brings a variety of important problems, concerning protection of the intellectual and production rights, that badly need to be solved.

DIGITAL STEGANOGRAPHY and DIGITAL WATERMARJIN

A very important property of (digital) information is that it is, in principle, very easy to produce and distribute unlimited number of its copies.

This might undermine the music, film, book and software industries and therefore it brings a variety of important problems, concerning protection of the intellectual and production rights, that badly need to be solved.

Since an unlimited number of perfect copies of text, audio and video data can be illegally produced and distributed requires to develop ways of embedding copyright and source information in audio and video data.

DIGITAL STEGANOGRAPHY and DIGITAL WATERMARJIN

A very important property of (digital) information is that it is, in principle, very easy to produce and distribute unlimited number of its copies.

This might undermine the music, film, book and software industries and therefore it brings a variety of important problems, concerning protection of the intellectual and production rights, that badly need to be solved.

Since an unlimited number of perfect copies of text, audio and video data can be illegally produced and distributed requires to develop ways of embedding copyright and source information in audio and video data.

Digital steganography and digital watermarking bring techniques to hide important information, in an undetectable and/or irremovable way, in audio and video digital data.

DIGITAL STEGANOGRAPHY and DIGITAL WATERMARJIN

A very important property of (digital) information is that it is, in principle, very easy to produce and distribute unlimited number of its copies.

This might undermine the music, film, book and software industries and therefore it brings a variety of important problems, concerning protection of the intellectual and production rights, that badly need to be solved.

Since an unlimited number of perfect copies of text, audio and video data can be illegally produced and distributed requires to develop ways of embedding copyright and source information in audio and video data.

Digital steganography and digital watermarking bring techniques to hide important information, in an undetectable and/or irremovable way, in audio and video digital data.

Digital steganography is the art and science of embedding information/signals in such a hidden way, especially in texts, images, video and audio carriers, that only intended recipients can recover them.

DIGITAL STEGANOGRAPHY and DIGITAL WATERMARJIN

A very important property of (digital) information is that it is, in principle, very easy to produce and distribute unlimited number of its copies.

This might undermine the music, film, book and software industries and therefore it brings a variety of important problems, concerning protection of the intellectual and production rights, that badly need to be solved.

Since an unlimited number of perfect copies of text, audio and video data can be illegally produced and distributed requires to develop ways of embedding copyright and source information in audio and video data.

Digital steganography and digital watermarking bring techniques to hide important information, in an undetectable and/or irremovable way, in audio and video digital data.

Digital steganography is the art and science of embedding information/signals in such a hidden way, especially in texts, images, video and audio carriers, that only intended recipients can recover them.

Digital watermarking is a process of embedding (hiding) information (through "watermarks") into digital data (signals) - picture, audio or video - to identify its owner or to authentisized its origin in an unremovable way.

DIGITAL STEGANOGRAPHY and DIGITAL WATERMARJIN

A very important property of (digital) information is that it is, in principle, very easy to produce and distribute unlimited number of its copies.

This might undermine the music, film, book and software industries and therefore it brings a variety of important problems, concerning protection of the intellectual and production rights, that badly need to be solved.

Since an unlimited number of perfect copies of text, audio and video data can be illegally produced and distributed requires to develop ways of embedding copyright and source information in audio and video data.

Digital steganography and digital watermarking bring techniques to hide important information, in an undetectable and/or irremovable way, in audio and video digital data.

Digital steganography is the art and science of embedding information/signals in such a hidden way, especially in texts, images, video and audio carriers, that only intended recipients can recover them.

Digital watermarking is a process of embedding (hiding) information (through "watermarks") into digital data (signals) - picture, audio or video - to identify its owner or to authentisized its origin in an unremovable way.

INFORMATION HIDING SUB-DISCIPLINES

Covert channels occur especially in operating systems and networks. They are communication paths that were neither designed nor intended to transfer information at all, but can be used that way.

These channels are typically used by untrustworthy/spying programs to leak (confidential) information to their owner while performing service for another user/program.

INFORMATION HIDING SUB-DISCIPLINES

Covert channels occur especially in operating systems and networks. They are communication paths that were neither designed nor intended to transfer information at all, but can be used that way.

These channels are typically used by untrustworthy/spying programs to leak (confidential) information to their owner while performing service for another user/program.

Anonymity is finding ways to hide meta content of the message (for example who is the sender and/or the recipients of a message). Anonymity is needed, for example, when making on-line voting, or to hide access to some web pages, or to hide sender.

INFORMATION HIDING SUB-DISCIPLINES

Covert channels occur especially in operating systems and networks. They are communication paths that were neither designed nor intended to transfer information at all, but can be used that way.

These channels are typically used by untrustworthy/spying programs to leak (confidential) information to their owner while performing service for another user/program.

Anonymity is finding ways to hide meta content of the message (for example who is the sender and/or the recipients of a message). Anonymity is needed, for example, when making on-line voting, or to hide access to some web pages, or to hide sender.

Steganography – covered writing – from Greek $\sigma\tau\varepsilon\gamma\alpha\nu$–$\xi$ $\gamma\rho\alpha\phi$–$\varepsilon\iota\nu$

is the art and science of hiding secret messages in innocently looking ones.

INFORMATION HIDING SUB-DISCIPLINES

Covert channels occur especially in operating systems and networks. They are communication paths that were neither designed nor intended to transfer information at all, but can be used that way.

These channels are typically used by untrustworthy/spying programs to leak (confidential) information to their owner while performing service for another user/program.

Anonymity is finding ways to hide meta content of the message (for example who is the sender and/or the recipients of a message). Anonymity is needed, for example, when making on-line voting, or to hide access to some web pages, or to hide sender.

Steganography – covered writing – from Greek $\sigma\tau\varepsilon\gamma\alpha\nu$–$\xi$ $\gamma\rho\alpha\phi$–$\varepsilon\iota\nu$

is the art and science of hiding secret messages in innocently looking ones.

Watermarking – is the technique to embed visible and especially imperceptible (invisible, transparent,...) watermarks into carriers in undetectable or unremovable way.

STEGANOGRAPHY versus WATERMARKING.II

Both techniques belong to the category of information hiding, but the objectives and embeddings of these techniques are just opposite.

STEGANOGRAPHY versus WATERMARKING.II

Both techniques belong to the category of information hiding, but the objectives and embeddings of these techniques are just opposite.

In watermarking, the important information is in the cover data. The embedded data - watermarks - are for protection or detection of the cover data origins.

STEGANOGRAPHY versus WATERMARKING.II

Both techniques belong to the category of information hiding, but the objectives and embeddings of these techniques are just opposite.

In watermarking, the important information is in the cover data. The embedded data - watermarks - are for protection or detection of the cover data origins.

In steganography, the cover data is not important. It mostly serves as a diversion from the most important information that is in embedded data.

Comment Steganography tools typically embed/hide relatively large blocks of information while watermarking tools embed/hide less information in an image or sounds or videos or texts.

Data hiding dilemma: to find the best trade-off between three quantities of embeddings: robustness, capacity and security.

STEGANOGRAPHY versus WATERMARKING again

Technically, differences between steganography and watermarking are both subtle and quite essential.

STEGANOGRAPHY versus WATERMARKING again

Technically, differences between steganography and watermarking are both subtle and quite essential.

The main goal of steganography is to hide a message m in some audio or video (cover) data d, to obtain new data d', in such a way that an eavesdropper **cannot detect** the presence of m in d'.

STEGANOGRAPHY versus WATERMARKING again

Technically, differences between steganography and watermarking are both subtle and quite essential.

The main goal of steganography is to hide a message m in some audio or video (cover) data d, to obtain new data d', in such a way that an eavesdropper **cannot detect** the presence of m in d'.

The main goal of watermarking is to hide a message m in some audio or video (cover) data d, to obtain new data d', practically indistinguishable from d, by people, in such a way that an eavesdropper **cannot remove or replace m** in d'.

STEGANOGRAPHY versus WATERMARKING again

Technically, differences between steganography and watermarking are both subtle and quite essential.

The main goal of steganography is to hide a message m in some audio or video (cover) data d, to obtain new data d', in such a way that an eavesdropper **cannot detect** the presence of m in d'.

The main goal of watermarking is to hide a message m in some audio or video (cover) data d, to obtain new data d', practically indistinguishable from d, by people, in such a way that an eavesdropper **cannot remove or replace m** in d'.

Shortly, one can say that cryptography is about **protecting** the content of messages, steganography is about **concealing** its very existence.

STEGANOGRAPHY versus WATERMARKING again

Technically, differences between steganography and watermarking are both subtle and quite essential.

The main goal of steganography is to hide a message m in some audio or video (cover) data d, to obtain new data d', in such a way that an eavesdropper **cannot detect** the presence of m in d'.

The main goal of watermarking is to hide a message m in some audio or video (cover) data d, to obtain new data d', practically indistinguishable from d, by people, in such a way that an eavesdropper **cannot remove or replace m** in d'.

Shortly, one can say that cryptography is about **protecting** the content of messages, steganography is about **concealing** its very existence.

Steganography methods usually do not need to provide strong security against removing or modification of the hidden message. Watermarking methods need to to be very robust to attempts to remove or modify a hidden message.

BASIC QUESTIONS

- Where and how can be secret data undetectably hidden?

BASIC QUESTIONS

- Where and how can be secret data undetectably hidden?
- Who and why needs steganography or watermarking?
- What is the maximum amount of information that can be hidden, given a level of degradation, to the digital media?
- How one chooses good cover media for a given stego message?
- How to detect, localize a stego message?

SOME APPLICATIONS of STEGANOGRAPHY

- To have secure secret communications where cryptographic encryption methods are not available.

SOME APPLICATIONS of STEGANOGRAPHY

- To have secure secret communications where cryptographic encryption methods are not available.
- To have secure secret communication where strong cryptography is impossible.

SOME APPLICATIONS of STEGANOGRAPHY

- To have secure secret communications where cryptographic encryption methods are not available.
- To have secure secret communication where strong cryptography is impossible.
- In some cases, for example in military applications, even the knowledge that two parties communicate can be of large importance.

SOME APPLICATIONS of STEGANOGRAPHY

- To have secure secret communications where cryptographic encryption methods are not available.
- To have secure secret communication where strong cryptography is impossible.
- In some cases, for example in military applications, even the knowledge that two parties communicate can be of large importance.
- The health care, and especially medical imaging systems, may very much benefit from information hiding techniques.

SOME APPLICATIONS of WATERMARKING

A basic application of watermarking techniques is to provide ownership information of digital data (images, video and audio products) by embedding copyright information into them.

SOME APPLICATIONS of WATERMARKING

A basic application of watermarking techniques is to provide ownership information of digital data (images, video and audio products) by embedding copyright information into them.

Other applications:

SOME APPLICATIONS of WATERMARKING

A basic application of watermarking techniques is to provide ownership information of digital data (images, video and audio products) by embedding copyright information into them.

Other applications:

- Automatic monitoring and tracking of copy-write material on WEB. (For example, a robot searches the Web for marked material and thereby identifies potential illegal issues.)

SOME APPLICATIONS of WATERMARKING

A basic application of watermarking techniques is to provide ownership information of digital data (images, video and audio products) by embedding copyright information into them.

Other applications:

- Automatic monitoring and tracking of copy-write material on WEB. (For example, a robot searches the Web for marked material and thereby identifies potential illegal issues.)
- Automatic audit of radio transmissions: (A robot can "listen" to a radio station and look for marks, which indicate that a particular piece of music, or advertisement , has been broadcast.)

SOME APPLICATIONS of WATERMARKING

A basic application of watermarking techniques is to provide ownership information of digital data (images, video and audio products) by embedding copyright information into them.

Other applications:

- Automatic monitoring and tracking of copy-write material on WEB. (For example, a robot searches the Web for marked material and thereby identifies potential illegal issues.)
- Automatic audit of radio transmissions: (A robot can “listen” to a radio station and look for marks, which indicate that a particular piece of music, or advertisement , has been broadcast.)
- Data augmentation – to add information for the benefit of the public.

SOME APPLICATIONS of WATERMARKING

A basic application of watermarking techniques is to provide ownership information of digital data (images, video and audio products) by embedding copyright information into them.

Other applications:

- Automatic monitoring and tracking of copy-write material on WEB. (For example, a robot searches the Web for marked material and thereby identifies potential illegal issues.)
- Automatic audit of radio transmissions: (A robot can "listen" to a radio station and look for marks, which indicate that a particular piece of music, or advertisement , has been broadcast.)
- Data augmentation – to add information for the benefit of the public.
- Fingerprinting applications (in order to distinguish distributed data)

Actually, watermarking has recently emerged as the leading technology to solve the above very important problems.

SOME APPLICATIONS of WATERMARKING

A basic application of watermarking techniques is to provide ownership information of digital data (images, video and audio products) by embedding copyright information into them.

Other applications:

- Automatic monitoring and tracking of copy-write material on WEB. (For example, a robot searches the Web for marked material and thereby identifies potential illegal issues.)
- Automatic audit of radio transmissions: (A robot can "listen" to a radio station and look for marks, which indicate that a particular piece of music, or advertisement , has been broadcast.)
- Data augmentation – to add information for the benefit of the public.
- Fingerprinting applications (in order to distinguish distributed data)

Actually, watermarking has recently emerged as the leading technology to solve the above very important problems.

All kind of data can be watermarked: audio, images, video, formatted text, 3D models, . . .

STEGANOGRAPHY/WATERMARKING versus CRYPTOGRAPHY

The purpose of both is to provide secret communication.

STEGANOGRAPHY/WATERMARKING versus CRYPTOGRAPHY

The purpose of both is to provide secret communication.

Cryptography hides the contents of the message from an attacker, but not the existence of the message.

STEGANOGRAPHY/WATERMARKING versus CRYPTOGRAPHY

The purpose of both is to provide secret communication.

Cryptography hides the contents of the message from an attacker, but not the existence of the message.

Steganography/watermarking even hide the very existence of the message in the communicated data.

STEGANOGRAPHY/WATERMARKING versus CRYPTOGRAPHY

The purpose of both is to provide secret communication.

Cryptography hides the contents of the message from an attacker, but not the existence of the message.

Steganography/watermarking even hide the very existence of the message in the communicated data.

Consequently, **the concept of breaking the system** is different for **cryptosystems** and **stegosystems (watermarking systems).**

STEGANOGRAPHY/WATERMARKING versus CRYPTOGRAPHY

The purpose of both is to provide secret communication.

Cryptography hides the contents of the message from an attacker, but not the existence of the message.

Steganography/watermarking even hide the very existence of the message in the communicated data.

Consequently, **the concept of breaking the system** is different for **cryptosystems** and **stegosystems (watermarking systems).**

- A cryptographic system is broken when the attacker can read the secrete message.

STEGANOGRAPHY/WATERMARKING versus CRYPTOGRAPHY

The purpose of both is to provide secret communication.

Cryptography hides the contents of the message from an attacker, but not the existence of the message.

Steganography/watermarking even hide the very existence of the message in the communicated data.

Consequently, **the concept of breaking the system** is different for **cryptosystems** and **stegosystems (watermarking systems).**

- A cryptographic system is broken when the attacker can read the secrete message.
- Breaking of a steganographic/watermarking system has two stages:
  - The attacker can detect that steganography/watermarking has been used;

STEGANOGRAPHY/WATERMARKING versus CRYPTOGRAPHY

The purpose of both is to provide secret communication.

Cryptography hides the contents of the message from an attacker, but not the existence of the message.

Steganography/watermarking even hide the very existence of the message in the communicated data.

Consequently, **the concept of breaking the system** is different for **cryptosystems** and **stegosystems (watermarking systems).**

- A cryptographic system is broken when the attacker can read the secrete message.
- Breaking of a steganographic/watermarking system has two stages:
  - The attacker can detect that steganography/watermarking has been used;
  - The attacker is able to read, modify or remove the hidden message.

STEGANOGRAPHY/WATERMARKING versus CRYPTOGRAPHY

The purpose of both is to provide secret communication.

Cryptography hides the contents of the message from an attacker, but not the existence of the message.

Steganography/watermarking even hide the very existence of the message in the communicated data.

Consequently, **the concept of breaking the system** is different for **cryptosystems** and **stegosystems (watermarking systems).**

- A cryptographic system is broken when the attacker can read the secrete message.
- Breaking of a steganographic/watermarking system has two stages:
  - The attacker can detect that steganography/watermarking has been used;
  - The attacker is able to read, modify or remove the hidden message.

A steganography/watermarking system is considered as insecure already if the detection of steganography/watermarking is possible.

STEGANOGRAPHY/WATERMARKING versus CRYPTOGRAPHY

The purpose of both is to provide secret communication.

Cryptography hides the contents of the message from an attacker, but not the existence of the message.

Steganography/watermarking even hide the very existence of the message in the communicated data.

Consequently, **the concept of breaking the system** is different for **cryptosystems** and **stegosystems (watermarking systems).**

- A cryptographic system is broken when the attacker can read the secrete message.
- Breaking of a steganographic/watermarking system has two stages:
    - The attacker can detect that steganography/watermarking has been used;
    - The attacker is able to read, modify or remove the hidden message.

A steganography/watermarking system is considered as insecure already if the detection of steganography/watermarking is possible.

The advantage of steganography over cryptography is that messages do not attract attention to themselves.

Steganography can be also use to increase secrecy provided by cryptographical methods

Steganography can be also use to increase secrecy provided by cryptographical methods

Indeed, when steganography is used to hide the encrypted communication, an enemy is not only faced with a difficult decryption problem, but also with the problem of finding the communicated data.

FIRST STEGANOGRAPHIC METHODS

- First recorded use of steganographic methods was traced to 440 BC. Greek Demaratus sent a warning about an attack by writing it on a wooden desk and then covering it by vax.

FIRST STEGANOGRAPHIC METHODS

- First recorded use of steganographic methods was traced to 440 BC. Greek Demaratus sent a warning about an attack by writing it on a wooden desk and then covering it by vax.
- Ancient Chinese wrote messages on fine silk, which was then crunched into a tiny ball and covered in wax. The messenger then swallowed the ball of wax.

FIRST STEGANOGRAPHIC METHODS

- First recorded use of steganographic methods was traced to 440 BC. Greek Demaratus sent a warning about an attack by writing it on a wooden desk and then covering it by vax.
- Ancient Chinese wrote messages on fine silk, which was then crunched into a tiny ball and covered in wax. The messenger then swallowed the ball of wax.
- A variety of steganographic methods was used also in Roman times and then in 15-16 century (ranging from coding messages in music, and string knots, to invisible inks).

FIRST STEGANOGRAPHIC METHODS

- First recorded use of steganographic methods was traced to 440 BC. Greek Demaratus sent a warning about an attack by writing it on a wooden desk and then covering it by vax.
- Ancient Chinese wrote messages on fine silk, which was then crunched into a tiny ball and covered in wax. The messenger then swallowed the ball of wax.
- A variety of steganographic methods was used also in Roman times and then in 15-16 century (ranging from coding messages in music, and string knots, to invisible inks).
- In the sixteenth century, the Italian scientist Giovanni Porta described how to conceal a message within a hard-boiled egg by making an ink from a mixture of one ounce of alum and a pint of vinegar, and then using ink to write on the shell. The ink penetrated the porous shell, and left the message on the surface of the hardened egg albumen, which could be read only when the shell was removed.

FIRST STEGANOGRAPHIC METHODS

- First recorded use of steganographic methods was traced to 440 BC. Greek Demaratus sent a warning about an attack by writing it on a wooden desk and then covering it by vax.
- Ancient Chinese wrote messages on fine silk, which was then crunched into a tiny ball and covered in wax. The messenger then swallowed the ball of wax.
- A variety of steganographic methods was used also in Roman times and then in 15-16 century (ranging from coding messages in music, and string knots, to invisible inks).
- In the sixteenth century, the Italian scientist Giovanni Porta described how to conceal a message within a hard-boiled egg by making an ink from a mixture of one ounce of alum and a pint of vinegar, and then using ink to write on the shell. The ink penetrated the porous shell, and left the message on the surface of the hardened egg albumen, which could be read only when the shell was removed.
- Special invisible "inks" (milk, urine,...) were important steganographic tools since middle ages and even during the Second World War.

FIRST STEGANOGRAPHIC METHODS

- First recorded use of steganographic methods was traced to 440 BC. Greek Demaratus sent a warning about an attack by writing it on a wooden desk and then covering it by vax.
- Ancient Chinese wrote messages on fine silk, which was then crunched into a tiny ball and covered in wax. The messenger then swallowed the ball of wax.
- A variety of steganographic methods was used also in Roman times and then in 15-16 century (ranging from coding messages in music, and string knots, to invisible inks).
- In the sixteenth century, the Italian scientist Giovanni Porta described how to conceal a message within a hard-boiled egg by making an ink from a mixture of one ounce of alum and a pint of vinegar, and then using ink to write on the shell. The ink penetrated the porous shell, and left the message on the surface of the hardened egg albumen, which could be read only when the shell was removed.
- Special invisible "inks" (milk, urine,...) were important steganographic tools since middle ages and even during the Second World War.
- Acrostic - hiding messages in first, last or other letters of words was popular steganographic method since middle ages.

FIRST STEGANOGRAPHIC METHODS

- First recorded use of steganographic methods was traced to 440 BC. Greek Demaratus sent a warning about an attack by writing it on a wooden desk and then covering it by vax.
- Ancient Chinese wrote messages on fine silk, which was then crunched into a tiny ball and covered in wax. The messenger then swallowed the ball of wax.
- A variety of steganographic methods was used also in Roman times and then in 15-16 century (ranging from coding messages in music, and string knots, to invisible inks).
- In the sixteenth century, the Italian scientist Giovanni Porta described how to conceal a message within a hard-boiled egg by making an ink from a mixture of one ounce of alum and a pint of vinegar, and then using ink to write on the shell. The ink penetrated the porous shell, and left the message on the surface of the hardened egg albumen, which could be read only when the shell was removed.
- Special invisible "inks" (milk, urine,...) were important steganographic tools since middle ages and even during the Second World War.
- Acrostic - hiding messages in first, last or other letters of words was popular steganographic method since middle ages.
- During the Second World War a technique was developed to shrink photographically a page of text into a dot less than one millimeter in diameter, and then hide this microdot in an apparently innocuous letter. (The first microdot has been spotted by FBI in 1941.)

HISTORY of MICRODOTS

- In 1857, Brewster suggested hiding secret messages "in spaces not larger than a full stop or small dot of ink".

HISTORY of MICRODOTS

- In 1857, Brewster suggested hiding secret messages "in spaces not larger than a full stop or small dot of ink".
- In 1860 the problem of making tiny images was solved by French photographer Dragon.

HISTORY of MICRODOTS

- In 1857, Brewster suggested hiding secret messages "in spaces not larger than a full stop or small dot of ink".
- In 1860 the problem of making tiny images was solved by French photographer Dragon.
- During Franco-Prussian war (1870-1881) from besieged Paris messages were sent on microfilms using pigeon post.

HISTORY of MICRODOTS

- In 1857, Brewster suggested hiding secret messages "in spaces not larger than a full stop or small dot of ink".
- In 1860 the problem of making tiny images was solved by French photographer Dragon.
- During Franco-Prussian war (1870-1881) from besieged Paris messages were sent on microfilms using pigeon post.
- During the Russo-Japanese war (1905) microscopic images were hidden in ears, nostrils, and under fingernails.

HISTORY of MICRODOTS

- In 1857, Brewster suggested hiding secret messages "in spaces not larger than a full stop or small dot of ink".
- In 1860 the problem of making tiny images was solved by French photographer Dragon.
- During Franco-Prussian war (1870-1881) from besieged Paris messages were sent on microfilms using pigeon post.
- During the Russo-Japanese war (1905) microscopic images were hidden in ears, nostrils, and under fingernails.
- During the First World War messages to and from spies were reduced to microdots, by several stages of photographic reductions, and then stuck on top of printed periods or commas (in innocuous cover materials, such as magazines).

FIRST STEGANOGRAPHY BOOKS

In the fourth century BC, the Greek Aeneas Tacticus, wrote a book on military techniques, On the defence of fortification in which the whole chapter is devoted to steganographic methods.

FIRST STEGANOGRAPHY BOOKS

In the fourth century BC, the Greek Aeneas Tacticus, wrote a book on military techniques, On the defence of fortification in which the whole chapter is devoted to steganographic methods.

In 1499 Johannes Trithemius, opat from Würzburg, wrote 3 out of 8 planned books "Steganographie".

FIRST STEGANOGRAPHY BOOKS

In the fourth century BC, the Greek Aeneas Tacticus, wrote a book on military techniques, On the defence of fortification in which the whole chapter is devoted to steganographic methods.

In 1499 Johannes Trithemius, opat from Würzburg, wrote 3 out of 8 planned books "Steganographie".

In 1518 Trithemius printed 6 books, 540 pages, on cryptography and steganography called **Polygraphiae.**

FIRST STEGANOGRAPHY BOOKS

In the fourth century BC, the Greek Aeneas Tacticus, wrote a book on military techniques, On the defence of fortification in which the whole chapter is devoted to steganographic methods.

In 1499 Johannes Trithemius, opat from Würzburg, wrote 3 out of 8 planned books "Steganographie".

In 1518 Trithemius printed 6 books, 540 pages, on cryptography and steganography called **Polygraphiae.**

This is Trithemius' most notorious work. It includes a sophisticated system of steganography, as well as angel magic. It also contains a synthesis of the science of knowledge, the art of memory, magic, an accelerated language learning system, and a method of sending messages without symbols.

FIRST STEGANOGRAPHY BOOKS

In the fourth century BC, the Greek Aeneas Tacticus, wrote a book on military techniques, On the defence of fortification in which the whole chapter is devoted to steganographic methods.

In 1499 Johannes Trithemius, opat from Würzburg, wrote 3 out of 8 planned books "Steganographie".

In 1518 Trithemius printed 6 books, 540 pages, on cryptography and steganography called **Polygraphiae.**

This is Trithemius' most notorious work. It includes a sophisticated system of steganography, as well as angel magic. It also contains a synthesis of the science of knowledge, the art of memory, magic, an accelerated language learning system, and a method of sending messages without symbols.

In 1665 Gaspari Schotti published the book "Steganographica", 400pages. (New presentation of Trithemius.)

TRITHEMIUS

- Born on February 2, 1462 and considered as one of the main intellectuals of his time.

TRITHEMIUS

- Born on February 2, 1462 and considered as one of the main intellectuals of his time.
- His book STEGANOGRAPHIA was published in 1606.

TRITHEMIUS

- Born on February 2, 1462 and considered as one of the main intellectuals of his time.
- His book STEGANOGRAPHIA was published in 1606.
- In 1609 catholic church has put the book on the list of forbidden books (to be there for more than 200 years).

TRITHEMIUS

- Born on February 2, 1462 and considered as one of the main intellectuals of his time.
- His book STEGANOGRAPHIA was published in 1606.
- In 1609 catholic church has put the book on the list of forbidden books (to be there for more than 200 years).
- His books are obscured by his strong belief in occult powers.

TRITHEMIUS

- Born on February 2, 1462 and considered as one of the main intellectuals of his time.
- His book STEGANOGRAPHIA was published in 1606.
- In 1609 catholic church has put the book on the list of forbidden books (to be there for more than 200 years).
- His books are obscured by his strong belief in occult powers.
- He classified witches into four categories.

TRITHEMIUS

- Born on February 2, 1462 and considered as one of the main intellectuals of his time.
- His book STEGANOGRAPHIA was published in 1606.
- In 1609 catholic church has put the book on the list of forbidden books (to be there for more than 200 years).
- His books are obscured by his strong belief in occult powers.
- He classified witches into four categories.
- He fixed creation of the world at 5206 B.C.

TRITHEMIUS

- Born on February 2, 1462 and considered as one of the main intellectuals of his time.
- His book STEGANOGRAPHIA was published in 1606.
- In 1609 catholic church has put the book on the list of forbidden books (to be there for more than 200 years).
- His books are obscured by his strong belief in occult powers.
- He classified witches into four categories.
- He fixed creation of the world at 5206 B.C.
- He described how to perform telepathy.

TRITHEMIUS

- Born on February 2, 1462 and considered as one of the main intellectuals of his time.
- His book STEGANOGRAPHIA was published in 1606.
- In 1609 catholic church has put the book on the list of forbidden books (to be there for more than 200 years).
- His books are obscured by his strong belief in occult powers.
- He classified witches into four categories.
- He fixed creation of the world at 5206 B.C.
- He described how to perform telepathy.
- Trithemius died on December 14, 1516.

TRITHEMIUS

- Born on February 2, 1462 and considered as one of the main intellectuals of his time.
- His book STEGANOGRAPHIA was published in 1606.
- In 1609 catholic church has put the book on the list of forbidden books (to be there for more than 200 years).
- His books are obscured by his strong belief in occult powers.
- He classified witches into four categories.
- He fixed creation of the world at 5206 B.C.
- He described how to perform telepathy.
- Trithemius died on December 14, 1516.

ORIGIN of MODERN - DIGITAL - STEGANOGRAPHY

The origin of modern (digital) steganography has been dated to around 1985 - after personal computers started to be applied to classical steganographic problems.

ORIGIN of MODERN - DIGITAL - STEGANOGRAPHY

The origin of modern (digital) steganography has been dated to around 1985 - after personal computers started to be applied to classical steganographic problems.

This was related to new problems at which information needed to be sent securely and safely between parties across restrictive communication channels.

B. Morgen and M. Bary, from a small Dallas based company created and fielded two steganographic systems.

ORIGIN of MODERN - DIGITAL - STEGANOGRAPHY

The origin of modern (digital) steganography has been dated to around 1985 - after personal computers started to be applied to classical steganographic problems.

This was related to new problems at which information needed to be sent securely and safely between parties across restrictive communication channels.

B. Morgen and M. Bary, from a small Dallas based company created and fielded two steganographic systems.

Since then a huge spectrum of methods and tools have been discovered and developed for digital cryptography.

ORIGIN of MODERN - DIGITAL - STEGANOGRAPHY

The origin of modern (digital) steganography has been dated to around 1985 - after personal computers started to be applied to classical steganographic problems.

This was related to new problems at which information needed to be sent securely and safely between parties across restrictive communication channels.

B. Morgen and M. Bary, from a small Dallas based company created and fielded two steganographic systems.

Since then a huge spectrum of methods and tools have been discovered and developed for digital cryptography.

Some examples"

- Network steganohraphy
- WLAN steganography
- Inter-protocol steganography
- Blog steganography
- Echo steganography
- Sudoku puzzles using steganography

Steganography used before is usually called physical steganography because physical carrier have been used to embed secret messages.

GENERAL STEGANOGRAPHIC MODEL

A general model of a steganographic system:

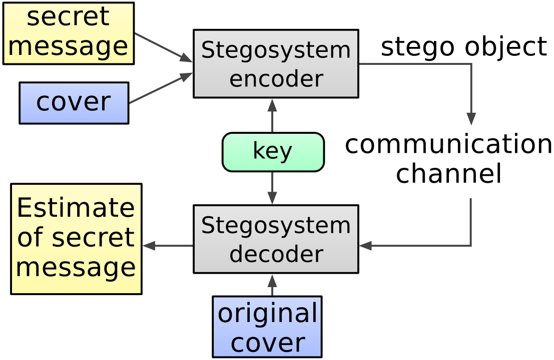


Figure 1: Model of steganographic systems

Steganographic algorithms are in general based on replacing noise component of a digital object with a to-be-hidden message.

Kerckhoffs's principle holds also for steganography. Security of the system should not be based on hiding embedding algorithm, but on hiding the key.

BASIC CONCEPTS of STEGOSYSTEMS

- Covertext (cover-data – cover-object) is an original (unaltered) message.

BASIC CONCEPTS of STEGOSYSTEMS

- Covertext (cover-data – cover-object) is an original (unaltered) message.
- Embedding process (ukryvaci process) in which the sender, Alice, tries to hide a message by embedding it into a (randomly chosen) covertext, usually using a key, to obtain a stegotext (stego-data or stego-object). The embedding process can be described by the mapping $E : C \times K \times M \rightarrow C$, where C is the set of possible cover – and stegotexts, K is the set of keys, and M is the set of messages.

BASIC CONCEPTS of STEGOSYSTEMS

- Covertext (cover-data – cover-object) is an original (unaltered) message.
- Embedding process (ukryvaci process) in which the sender, Alice, tries to hide a message by embedding it into a (randomly chosen) covertext, usually using a key, to obtain a stegotext (stego-data or stego-object). The embedding process can be described by the mapping $E : C \times K \times M \to C$, where C is the set of possible cover – and stegotexts, K is the set of keys, and M is the set of messages.
- Stegotext (stego-data – stego-object) is the message that comes out of the embedding process and contains the hidden message.

BASIC CONCEPTS of STEGOSYSTEMS

- Covertext (cover-data – cover-object) is an original (unaltered) message.
- Embedding process (ukryvaci process) in which the sender, Alice, tries to hide a message by embedding it into a (randomly chosen) covertext, usually using a key, to obtain a stegotext (stego-data or stego-object). The embedding process can be described by the mapping $E : C \times K \times M \rightarrow C$, where C is the set of possible cover – and stegotexts, K is the set of keys, and M is the set of messages.
- Stegotext (stego-data – stego-object) is the message that comes out of the embedding process and contains the hidden message.
- Recovering process (or extraction process – odkryvaci process) in which the receiver, Bob, tries to get, using the key only but not the covertext, the hidden message in the stegotext.

  The recovery (decoding) process D can be seen as a mapping $D : C \times K \rightarrow C$.

BASIC CONCEPTS of STEGOSYSTEMS

- Covertext (cover-data – cover-object) is an original (unaltered) message.
- Embedding process (ukryvaci process) in which the sender, Alice, tries to hide a message by embedding it into a (randomly chosen) covertext, usually using a key, to obtain a stegotext (stego-data or stego-object). The embedding process can be described by the mapping $E : C \times K \times M \rightarrow C$, where C is the set of possible cover – and stegotexts, K is the set of keys, and M is the set of messages.
- Stegotext (stego-data – stego-object) is the message that comes out of the embedding process and contains the hidden message.
- Recovering process (or extraction process – odkryvaci process) in which the receiver, Bob, tries to get, using the key only but not the covertext, the hidden message in the stegotext.

  The recovery (decoding) process D can be seen as a mapping $D : C \times K \rightarrow C$.
- Security requirement is that a third person watching such a communication should not be able to find out whether the sender has been active, and when, in the sense that he really embedded a message in the covertext. In other words, stegotexts should be indistinguishable from covertexts.

BASIC TYPES of STEGOSYSTEMS

There are three basic types of stegosystems

- Pure stegosystems – no key is used.
- Secret-key stegosystems – shared secret key is used.
- Public-key stegosystems – public and secret keys are used.

BASIC TYPES of STEGOSYSTEMS

There are three basic types of stegosystems

- Pure stegosystems – no key is used.
- Secret-key stegosystems – shared secret key is used.
- Public-key stegosystems – public and secret keys are used.

Definition **Pure stegosystem** $S = \langle C, M, E, D \rangle$, where C is the set of possible covertexts, M is the set of secret messages, $|C| \geq |M|$, $E : C \times M \rightarrow C$ is the embedding function and $D : C \rightarrow M$, is the extraction function, with the property that D(E(c,m)) = m, for all $m \in M$ and $c \in C$.

Security of the pure stegosystems depends completely on its secrecy.On the other hand, security of other two stegosystems depends on the secrecy of the key used.

BASIC TYPES of STEGOSYSTEMS

There are three basic types of stegosystems

- Pure stegosystems – no key is used.
- Secret-key stegosystems – shared secret key is used.
- Public-key stegosystems – public and secret keys are used.

Definition **Pure stegosystem** $S = \langle C, M, E, D \rangle$, where C is the set of possible covertexts, M is the set of secret messages, $|C| \geq |M|$, $E : C \times M \rightarrow C$ is the embedding function and $D : C \rightarrow M$, is the extraction function, with the property that D(E(c,m)) = m, for all $m \in M$ and $c \in C$.

Security of the pure stegosystems depends completely on its secrecy.On the other hand, security of other two stegosystems depends on the secrecy of the key used.

Definition **Secret-key (asymmetric) stegosystem** $S = \langle C, M, K, E_K, D_K \rangle$, where C is the set of possible covertexts, M is the set of secret messages with $|C| \geq |M|$, K is the set of secret keys, $E_K : C \times M \times K \rightarrow C$, $D_K : C \times K \rightarrow M$ with the property that $D_K(E_K(c, m, k), k) = m$ for all $m \in M$, $c \in C$ and $k \in K$.

PUBLIC-KEY STEGANOGRAPHY

Similarly as in the case of the public-key cryptography, two keys are used: a public-key E for embedding and a private-key D for recovering.

PUBLIC-KEY STEGANOGRAPHY

Similarly as in the case of the public-key cryptography, two keys are used: a public-key E for embedding and a private-key D for recovering.

It is often useful to combine such a public-key stegosystem with a public-key cryptosystem.

PUBLIC-KEY STEGANOGRAPHY

Similarly as in the case of the public-key cryptography, two keys are used: a public-key E for embedding and a private-key D for recovering.

It is often useful to combine such a public-key stegosystem with a public-key cryptosystem.

For example, in case Alice wants to send a message m to Bob, she encodes first m using Bob's public key e_B, then makes embedding of $e_B(m)$ using process E into a cover and then sends the resulting stegotext to Bob, who recovers $e_B(m)$ using D and then decrypts it, using his decryption function d_B.

TEXT STEGANOGRAPHY

A variety of steganography techniques allow to hide messages in formatted texts.

- Acrostic. A message is hidden into certain letters of the text, for example into the first letters of some words.

  Tables have been produced, the first one by Trithentius, called Ave Maria, how to replace plaintext letters by words.

TEXT STEGANOGRAPHY

A variety of steganography techniques allow to hide messages in formatted texts.

- Acrostic. A message is hidden into certain letters of the text, for example into the first letters of some words.

  Tables have been produced, the first one by Trithentius, called Ave Maria, how to replace plaintext letters by words.
- An improvement of the previous method is to distribute plaintext letters randomly in the cover-text and then use a mask to read it.

TEXT STEGANOGRAPHY

A variety of steganography techniques allow to hide messages in formatted texts.

- Acrostic. A message is hidden into certain letters of the text, for example into the first letters of some words.

  Tables have been produced, the first one by Trithentius, called Ave Maria, how to replace plaintext letters by words.
- An improvement of the previous method is to distribute plaintext letters randomly in the cover-text and then use a mask to read it.
- The presence of errors or stylistic features at predetermined points in the cover data is another way to select the location of the embedded information.

TEXT STEGANOGRAPHY

A variety of steganography techniques allow to hide messages in formatted texts.

- Acrostic. A message is hidden into certain letters of the text, for example into the first letters of some words.

  Tables have been produced, the first one by Trithentius, called Ave Maria, how to replace plaintext letters by words.
- An improvement of the previous method is to distribute plaintext letters randomly in the cover-text and then use a mask to read it.
- The presence of errors or stylistic features at predetermined points in the cover data is another way to select the location of the embedded information.
- Line shifting encodings.

TEXT STEGANOGRAPHY

A variety of steganography techniques allow to hide messages in formatted texts.

- Acrostic. A message is hidden into certain letters of the text, for example into the first letters of some words.

  Tables have been produced, the first one by Trithentius, called Ave Maria, how to replace plaintext letters by words.
- An improvement of the previous method is to distribute plaintext letters randomly in the cover-text and then use a mask to read it.
- The presence of errors or stylistic features at predetermined points in the cover data is another way to select the location of the embedded information.
- Line shifting encodings.
- Word shifting encodings.

TEXT STEGANOGRAPHY

A variety of steganography techniques allow to hide messages in formatted texts.

- Acrostic. A message is hidden into certain letters of the text, for example into the first letters of some words.

  Tables have been produced, the first one by Trithentius, called Ave Maria, how to replace plaintext letters by words.
- An improvement of the previous method is to distribute plaintext letters randomly in the cover-text and then use a mask to read it.
- The presence of errors or stylistic features at predetermined points in the cover data is another way to select the location of the embedded information.
- Line shifting encodings.
- Word shifting encodings.
- Data hiding through justifications.

TEXT STEGANOGRAPHY

A variety of steganography techniques allow to hide messages in formatted texts.

- Acrostic. A message is hidden into certain letters of the text, for example into the first letters of some words.

  Tables have been produced, the first one by Trithentius, called Ave Maria, how to replace plaintext letters by words.
- An improvement of the previous method is to distribute plaintext letters randomly in the cover-text and then use a mask to read it.
- The presence of errors or stylistic features at predetermined points in the cover data is another way to select the location of the embedded information.
- Line shifting encodings.
- Word shifting encodings.
- Data hiding through justifications.
- Through features encoding (for example in the vertical lines of letters b, d, h, k).

TEXT STEGANOGRAPHY

A variety of steganography techniques allow to hide messages in formatted texts.

- Acrostic. A message is hidden into certain letters of the text, for example into the first letters of some words.

  Tables have been produced, the first one by Trithentius, called Ave Maria, how to replace plaintext letters by words.
- An improvement of the previous method is to distribute plaintext letters randomly in the cover-text and then use a mask to read it.
- The presence of errors or stylistic features at predetermined points in the cover data is another way to select the location of the embedded information.
- Line shifting encodings.
- Word shifting encodings.
- Data hiding through justifications.
- Through features encoding (for example in the vertical lines of letters b, d, h, k).

Text steganography (a really good one) is considered to be very difficult kind of steganography due to the lack of redundancy in texts comparing to images or audio.

ACROSTIC

Amorosa visione by Giovanni Boccaccio (1313-1375) is said to be the **world largest acrostic.**

Boccaccio first wrote three sonnets (1500 letters together) and then he wrote other poems such that the initials of the successive tercets correspond exactly to the letters of the sonnets.

ACROSTIC

Amorosa visione by Giovanni Boccaccio (1313-1375) is said to be the **world largest acrostic.**

Boccaccio first wrote three sonnets (1500 letters together) and then he wrote other poems such that the initials of the successive tercets correspond exactly to the letters of the sonnets.

In the book **Hypnerotomachia Poliphili**, published **by an anonymous** in 1499, and considered as one of the most beautiful books ever,the first letters of the 38 chapters spelled out as follows:

Poliam frater Franciscus Columna peramavit

with the translation

Brother Francesco Colonna passionately loves Polia

PERFECT SECRECY of STEGOSYSTEMS

In order to define secrecy of a stegosystem we need to consider

- probability distribution P_C on the set C of covertexts;
- probability distribution P_M on the set M of secret messages;
- probability distribution P_K on the set K of keys;
- probability distribution P_S on the set $\{E_K(c, m, k), | c \in C, m \in M, k \in K\}$ of stegotexts.

PERFECT SECRECY of STEGOSYSTEMS

In order to define secrecy of a stegosystem we need to consider

- probability distribution P_C on the set C of covertexts;
- probability distribution P_M on the set M of secret messages;
- probability distribution P_K on the set K of keys;
- probability distribution P_S on the set $\{E_K(c, m, k), | c \in C, m \in M, k \in K\}$ of stegotexts.

The basic related concept is that of the relative entropy $D(P_1 \| P_2)$ of two probability distributions P_1 and P_2 defined on a set Q by

$$D(P_1 \| P_2) = \sum_{q \in Q} P_1(q) lg \frac{P_1(q)}{P_2(q)},$$

which measures the inefficiency of assuming that the distribution on Q is P_2 if it is really P_1.

PERFECT SECRECY of STEGOSYSTEMS

In order to define secrecy of a stegosystem we need to consider

- probability distribution P_C on the set C of covertexts;
- probability distribution P_M on the set M of secret messages;
- probability distribution P_K on the set K of keys;
- probability distribution P_S on the set $\{E_K(c,m,k),|c \in C, m \in M, k \in K\}$ of stegotexts.

The basic related concept is that of the relative entropy $D(P_1\|P_2)$ of two probability distributions P_1 and P_2 defined on a set Q by

$$D(P_1\|P_2) = \sum_{q\in Q} P_1(q) lg \frac{P_1(q)}{P_2(q)},$$

which measures the inefficiency of assuming that the distribution on Q is P_2 if it is really P_1.

Definition Let S be a stegosystem, P_C the probability distribution on covertexts C and P_S the probability distribution of the stegotexts and $\varepsilon > 0$. S is called – ε-secure against passive attackers, if

$$D(P_C\|P_S) \leq \varepsilon$$

and **perfectly secure** if $\varepsilon = 0$.

PERFECTLY SECURE STEGOSYSTEMS

A perfectly secure stegosystem can be constructed out of the ONE TIME-PAD CRYPTOSYSTEM

PERFECTLY SECURE STEGOSYSTEMS

A perfectly secure stegosystem can be constructed out of the ONE TIME-PAD CRYPTOSYSTEM

Theorem There exist perfectly secure stegosystems.

PERFECTLY SECURE STEGOSYSTEMS

A perfectly secure stegosystem can be constructed out of the ONE TIME-PAD CRYPTOSYSTEM

Theorem There exist perfectly secure stegosystems.

Proof. Let n be an integer, $C_n = \{0,1\}^n$ and P_C be the uniform distribution on C_n, and let $m \in C_n$ be a secret message.

The sender selects randomly $c \in C_n$, computes $c \oplus m = s$. The resulting stegotexts are uniformly distributed on C_n and therefore $P_C = P_S$ from what it follows that

$$D(P_{C_n} \| P_S) = 0.$$

In the extraction process, the message m can be extracted from s by the computation

$$m = s \oplus c.$$

INFORMATION HIDING in NOISY DATA

Perhaps the most basic methods of steganography is to utilize the existence of redundant information in communication channels/media.

INFORMATION HIDING in NOISY DATA

Perhaps the most basic methods of steganography is to utilize the existence of redundant information in communication channels/media.

Images and digital sounds naturally contain such redundancies in the form of noise components.

For images and digital sounds it is natural to assume that a cover-data are represented by a sequence of numbers and their least significant bits (LSB) represent noise.

INFORMATION HIDING in NOISY DATA

Perhaps the most basic methods of steganography is to utilize the existence of redundant information in communication channels/media.

Images and digital sounds naturally contain such redundancies in the form of noise components.

For images and digital sounds it is natural to assume that a cover-data are represented by a sequence of numbers and their least significant bits (LSB) represent noise.

If cover-data are represented by numbers

$$c_1, c_2, c_3, \ldots,$$

then one of the most basic steganographic methods is to replace, in some of c_i's, chosen using an algorithm and a key, the least significant bits by the bits of the message that should be hidden.

INFORMATION HIDING in NOISY DATA

Perhaps the most basic methods of steganography is to utilize the existence of redundant information in communication channels/media.

Images and digital sounds naturally contain such redundancies in the form of noise components.

For images and digital sounds it is natural to assume that a cover-data are represented by a sequence of numbers and their least significant bits (LSB) represent noise.

If cover-data are represented by numbers

$$c_1, c_2, c_3, \ldots,$$

then one of the most basic steganographic methods is to replace, in some of c_i's, chosen using an algorithm and a key, the least significant bits by the bits of the message that should be hidden.

Unfortunately, this method does not provide high level of security and it can change significantly statistical properties of the cover-data.

ACTIVE and MALICIOUS ATTACKS

At the design of stegosystems special attention has to be paid to the presence of active and malicious attackers.

- Active attackers can change cover during the communication process.
- An attacker is malicious if he forges messages or initiates a steganography protocol under the name of one communicating party.

ACTIVE and MALICIOUS ATTACKS

At the design of stegosystems special attention has to be paid to the presence of active and malicious attackers.

- Active attackers can change cover during the communication process.
- An attacker is malicious if he forges messages or initiates a steganography protocol under the name of one communicating party.

In the presence of a malicious attacker, it is not enough that stegosystem is robust.

If the embedding method does not depend on a key shared by the sender and receiver, then an attacker can forge messages, since the recipient is not able to verify sender's identity.

SECURITY of STEGOSYSTEMS

Definition A steganographic algorithm is called secure if

- Messages are hidden using a public algorithm and a secret key. The secret key must identify the sender uniquely.

Definition A steganographic algorithm is called secure if

- Messages are hidden using a public algorithm and a secret key. The secret key must identify the sender uniquely.
- Only the holder of the secret key can detect, extract and prove the existence of the hidden message. (Nobody else should be able to find any statistical evidence of a message's existence.)

SECURITY of STEGOSYSTEMS

Definition A steganographic algorithm is called secure if

- Messages are hidden using a public algorithm and a secret key. The secret key must identify the sender uniquely.
- Only the holder of the secret key can detect, extract and prove the existence of the hidden message. (Nobody else should be able to find any statistical evidence of a message's existence.)
- Even if an enemy gets the contents of one hidden message, he should have no chance of detecting others.

SECURITY of STEGOSYSTEMS

Definition A steganographic algorithm is called secure if

- Messages are hidden using a public algorithm and a secret key. The secret key must identify the sender uniquely.
- Only the holder of the secret key can detect, extract and prove the existence of the hidden message. (Nobody else should be able to find any statistical evidence of a message's existence.)
- Even if an enemy gets the contents of one hidden message, he should have no chance of detecting others.
- It is computationally infeasible to detect hidden messages.

STEGO – ATTACKS

Stego-only attack Only the stego-object is available for stegoanalysis.

Stego-only attack Only the stego-object is available for stegoanalysis.

Known-cover attack The original cover-object and stego-object are both available.

Stego-only attack Only the stego-object is available for stegoanalysis.

Known-cover attack The original cover-object and stego-object are both available.

Known-message attack Sometimes the hidden message may become known to the stegoanalyser. Analyzing the stego-object for patterns that correspond to the hidden message may be beneficial for future attacks against that system. (Even with the message, this may be very difficult and may even be considered equivalent to the stego-analysis.)

STEGO – ATTACKS

Stego-only attack Only the stego-object is available for stegoanalysis.

Known-cover attack The original cover-object and stego-object are both available.

Known-message attack Sometimes the hidden message may become known to the stegoanalyser. Analyzing the stego-object for patterns that correspond to the hidden message may be beneficial for future attacks against that system. (Even with the message, this may be very difficult and may even be considered equivalent to the stego-analysis.)

Chosen-stego attack The stegoanalysis generates a stego-object from some steganography tool or algorithm from a chosen message. The goal in this attack is to determine corresponding patterns in the stego-object that may point to the use of specific steganography tools or algorithms.

STEGO – ATTACKS

Stego-only attack Only the stego-object is available for stegoanalysis.

Known-cover attack The original cover-object and stego-object are both available.

Known-message attack Sometimes the hidden message may become known to the stegoanalyser. Analyzing the stego-object for patterns that correspond to the hidden message may be beneficial for future attacks against that system. (Even with the message, this may be very difficult and may even be considered equivalent to the stego-analysis.)

Chosen-stego attack The stegoanalysis generates a stego-object from some steganography tool or algorithm from a chosen message. The goal in this attack is to determine corresponding patterns in the stego-object that may point to the use of specific steganography tools or algorithms.

Known-stego attack The steganography algorithm is known and both the original and stego-objects are available.

Substitution techniques: substitute a redundant part of the cover-object with the secret message.

BASIC STEGANOGRAPHIC TECHNIQUES

Substitution techniques: substitute a redundant part of the cover-object with the secret message.

Transformed domain techniques: embed the secret message in a transform space of the signal (e.g. in the frequency domain).

BASIC STEGANOGRAPHIC TECHNIQUES

Substitution techniques: substitute a redundant part of the cover-object with the secret message.

Transformed domain techniques: embed the secret message in a transform space of the signal (e.g. in the frequency domain).

Spread spectrum techniques: embed the secret messages adopting ideas from the spread spectrum communications.

Substitution techniques: substitute a redundant part of the cover-object with the secret message.

Transformed domain techniques: embed the secret message in a transform space of the signal (e.g. in the frequency domain).

Spread spectrum techniques: embed the secret messages adopting ideas from the spread spectrum communications.

Statistical techniques: embed messages by changing some statistical properties of the cover-objects and use hypothesis-testing methods in the extraction process.

BASIC STEGANOGRAPHIC TECHNIQUES

Substitution techniques: substitute a redundant part of the cover-object with the secret message.

Transformed domain techniques: embed the secret message in a transform space of the signal (e.g. in the frequency domain).

Spread spectrum techniques: embed the secret messages adopting ideas from the spread spectrum communications.

Statistical techniques: embed messages by changing some statistical properties of the cover-objects and use hypothesis-testing methods in the extraction process.

Cover generation techniques: do not embed the message in randomly chosen cover-objects, but create covers that fit a message that needs to be hidden.

DIGITAL COVER DATA

A cover-object or, shortly, a cover c is a sequence of numbers $c_i, i = 1, 2, \ldots, |c|$.

Such a sequence can represent digital sounds in different time moments, or a linear (vectorized) version of an image.

$c_i \in \{0, 1\}$ in case of binary images and, usually, $0 \leq c_i \leq 256$ in case of quantized images or sounds.

DIGITAL COVER DATA

A cover-object or, shortly, a cover c is a sequence of numbers $c_i, i = 1, 2, \dots, |c|$.

Such a sequence can represent digital sounds in different time moments, or a linear (vectorized) version of an image.

$c_i \in \{0, 1\}$ in case of binary images and, usually, $0 \leq c_i \leq 256$ in case of quantized images or sounds.

An image C can be seen as a discrete function assigning a color vector c(x,y) to each pixel p(x,y).

DIGITAL COVER DATA

A cover-object or, shortly, a cover c is a sequence of numbers $c_i, i = 1, 2, \ldots, |c|$.

Such a sequence can represent digital sounds in different time moments, or a linear (vectorized) version of an image.

$c_i \in \{0, 1\}$ in case of binary images and, usually, $0 \leq c_i \leq 256$ in case of quantized images or sounds.

An image C can be seen as a discrete function assigning a color vector c(x,y) to each pixel p(x,y).

A color value is normally a three-component vector in a **color space**. Often used are the following color spaces:

RGB-space – every color is specified as a weighted sum of a red, green and a blue component. A vector specifies intensities of these three components.

YCbCr-space It distinguishes a luminance Y and two chrominance components (Cb, Cr).

Note A color vector can be converted to YCbCr components as follows:

$$\mathrm{Y} = 0.299\ \mathrm{R} + 0.587\ \mathrm{G} + 0.114\ \mathrm{B}$$
$$Cb = 0.5 + \frac{(B - Y)}{2}$$
$$Cr = 0.5 + \frac{(R - Y)}{1.6}$$

BASIC SUBSTITUTION TECHNIQUES

- LSB substitution – the LSB of an binary block c_{k_i} is replaced by the bit m_i of the secret message.

  The methods differ by techniques how to determine k_i for a given i.

  For example, $k_{i+1} = k_i + r_i$, where r_i is a sequence of numbers generated by a pseudo-random generator.

BASIC SUBSTITUTION TECHNIQUES

- LSB substitution – the LSB of an binary block c_{k_i} is replaced by the bit m_i of the secret message.

  The methods differ by techniques how to determine k_i for a given i.

  For example, $k_{i+1} = k_i + r_i$, where r_i is a sequence of numbers generated by a pseudo-random generator.
- Substitution into parity bits of blocks. If the parity bit of block c_{k_i} is m_i, then the block c_{k_i} is not changed; otherwise one of its bits is changed.
- Substitution in binary images. If image c_i has more (less) black pixels than white pixels and $m_i = 1 (m_i = 0)$, then c_i is not changed; otherwise the portion of black and white pixels is changed (by making changes at those pixels that are neighbors of pixels of the opposite color).
- Substitution in unused or reserved space in computer systems.

LSB SUBSTITUTION PLUSES and MINUSES

Bits for substitution can be chosen (a) randomly; (b) adaptively according to local properties of the digital media that is used.

Advantages:

(a) LSB substitution is the simplest and most common stego technique and it can be used also for different color models.

(b) This method can reach a very high capacity with little, if any, visible impact to the cover digital media.

(c) It is relatively easy to apply on images and radio data.

(d) Many tools for LSB substitutions are available on the internet

Disadvantages:

(a) It is relatively simple to detect the hidden data;

(b) It does not offer robustness against small modifications (including compression) at the stego images.

ROBUSTNESS of STEGANOGRAFY

Steganographic systems are extremely sensitive to cover modifications, such as

- image processing techniques (smoothing, filtering, image transformations, ...);
- filtering of digital sounds;
- compression techniques.

ROBUSTNESS of STEGANOGRAFY

Steganographic systems are extremely sensitive to cover modifications, such as

- image processing techniques (smoothing, filtering, image transformations, ...);
- filtering of digital sounds;
- compression techniques.

Informally, a stegosystem is **robust** if the embedded information cannot be altered without making substantial changes to the stego-objects.

Definition Let S be a stegosystem and P be a class of mappings $C \rightarrow C$. S is P-robust, if for all $p \in P$

$$D_K(p(E_K(c,m,k)),k) = D_K(E_K(c,m,k),k) = m$$

in the case of a secret-key stegosystem and

$$D(p(E(c,m))) = D(E(c,m)) = m$$

in the case of pure stegosystem, for any m, c, k.

- There is a clear tradeoff between *security* and *robustness*.
- Some stegosystems are designed to be robust against a specific class of mappings (for example JPEG compression/decompression).
- There are two basic approaches to make stegosystems robust:
  - By foreseeing possible cover modifications, the embedding process can be robust so that possible modifications do not entirely destroy embedded information.
  - Reversing operations that has been made by an active attacker.

DETECTING SECRET MESSAGES

The main goal of a passive attacker is to decide whether data sent to Bob by Alice contain secret message or not.

The detection task can be formalized as a statistical hypothesis-testing problem with the test function $f : C \rightarrow \{0, 1\}$:

$$f(c) = \begin{cases} 1, & \text{if } c \text{ contains a secret message;} \\ 0, & \text{otherwise} \end{cases}$$

There are two types of errors possible:

- Type-I error - a secret message is detected in data with no secret message;
- Type-II error - a hidden secret message is not detected

In the case of ε-secure stegosystems there is well know relation between the probability β of the type II error and probability α of the type I error.

Let S be a stegosystem which is ε-secure against passive attackers, β the probability that the attacker does not detect a hidden message and α the probability that the attacker falsely detects a hidden message. Then

$$d(\alpha, \beta) \leq \varepsilon,$$

where $d(\alpha, \beta)$ is the binary relative entropy defined by

$$d(\alpha, \beta) = \alpha \lg \frac{\alpha}{1 - \beta} + (1 - \alpha) \lg \frac{1 - \alpha}{\beta}.$$

DIGITAL WATERMARKING

Digital watermarking seems to be a promising technique to deal with the following problem:

Problem Digitalization allows to make unlimited number of copies of intellectual products (books, art products, music, video,...). How to make use of this enormous potential digitalization has and, at the same time, to protect intellectual rights of authors (copyrights, protection against modifications and insertion into other products), in a that is legally accepted?

Solution Digital watermarking tries to solve the above problem using a variety of methods of informatics, cryptography, signal processing, ... and in order to achieve that tries to insert specific information (watermarks) into data/carrier/signal in such a way that watermarks cannot be extracted or at least detected and if data with one or several watermarks are copied, watermarks should not change.

BASIC APPLICATIONS

- Copyright protection - ownership assertion For example, if a watermark is embedded into a music (or video) product, then each time music (video) is played in public information about author is extracted and tandem are established. Another example: annotation of digital photographs

BASIC APPLICATIONS

- Copyright protection - ownership assertion For example, if a watermark is embedded into a music (or video) product, then each time music (video) is played in public information about author is extracted and tandem are established. Another example: annotation of digital photographs
- Source tracing. Watermarks can be used to trace or verify the source of digital data.

BASIC APPLICATIONS

- Copyright protection - ownership assertion For example, if a watermark is embedded into a music (or video) product, then each time music (video) is played in public information about author is extracted and tandem are established. Another example: annotation of digital photographs
- Source tracing. Watermarks can be used to trace or verify the source of digital data.
- Insertion of additional (sensitive) information For example, personal data into röntgen photos r of keywords into multimedia products.

HISTORY of WATERMARKING

Paper watermarks appeared in the art of handmade paper marking 700 hundred years ago.

Watermarks were mainly used to identify the mill producing the paper and paper format, quality and strength.

Paper watermarks was a perfect technique to eliminate confusion from which mill paper is and what are its parameters.

HISTORY of WATERMARKING

Paper watermarks appeared in the art of handmade paper marking 700 hundred years ago.

Watermarks were mainly used to identify the mill producing the paper and paper format, quality and strength.

Paper watermarks was a perfect technique to eliminate confusion from which mill paper is and what are its parameters.

Legal power of watermarks has been demonstrated in 1887 in France when watermarks of two letters, presented as a piece of evidence in a trial, proved that the letters had been predated, what resulted in the downfall of a cabinet and, finally, the resignation of the president Grévy.

HISTORY of WATERMARKING

Paper watermarks appeared in the art of handmade paper marking 700 hundred years ago.

Watermarks were mainly used to identify the mill producing the paper and paper format, quality and strength.

Paper watermarks was a perfect technique to eliminate confusion from which mill paper is and what are its parameters.

Legal power of watermarks has been demonstrated in 1887 in France when watermarks of two letters, presented as a piece of evidence in a trial, proved that the letters had been predated, what resulted in the downfall of a cabinet and, finally, the resignation of the president Grévy.

Paper watermarks in bank notes or stamps inspired the first use of the term water mark in the context of digital data.

HISTORY of WATERMARKING

Paper watermarks appeared in the art of handmade paper marking 700 hundred years ago.

Watermarks were mainly used to identify the mill producing the paper and paper format, quality and strength.

Paper watermarks was a perfect technique to eliminate confusion from which mill paper is and what are its parameters.

Legal power of watermarks has been demonstrated in 1887 in France when watermarks of two letters, presented as a piece of evidence in a trial, proved that the letters had been predated, what resulted in the downfall of a cabinet and, finally, the resignation of the president Grévy.

Paper watermarks in bank notes or stamps inspired the first use of the term water mark in the context of digital data.

The first publications that really focused on watermarking of digital images were from 1990 and then in 1993.

in WATERMARKING SYSTEMS

Figure 2 shows the basic scheme of the watermarks embedding systems.

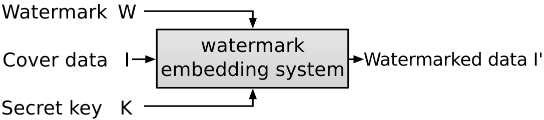


Figure 2: Watermark embedding scheme

Inputs to the scheme are the watermark, the cover data and an optional **public or secret** key. The output are watermarked data. The key is used to enforce security.

in WATERMARKING SYSTEMS

Figure 2 shows the basic scheme of the watermarks embedding systems.

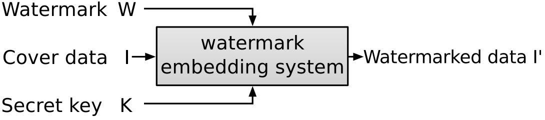


Figure 2: Watermark embedding scheme

Inputs to the scheme are the watermark, the cover data and an optional **public or secret** key. The output are watermarked data. The key is used to enforce security.

Figure 3 shows the basic scheme for watermark recovery schemes.

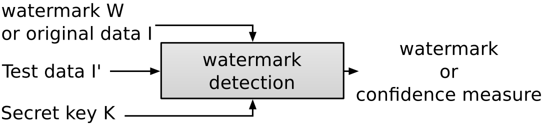


Figure 3: Watermark recovery scheme

Inputs to the scheme are the watermarked data, the **secret or public** key and, depending on the method, the original data **and/or** the original watermark. The output is the recovered watermark W or some kind of confidence measure indicating how likely it is for the given watermark at the input to be present in the data under inspection.

TYPES of WATERMARKING SCHEMES

Private (non-blind) watermarking systems require for extraction/detection the original cover-data.

- Type I systems use the original cover-data to determine where a watermark is and how to extract the watermark from stego-data.
- Type II systems require a copy of the embedded watermark for extraction and just yield a yes/no answer to the question whether the stego-data contains a watermark.

TYPES of WATERMARKING SCHEMES

Private (non-blind) watermarking systems require for extraction/detection the original cover-data.

- Type I systems use the original cover-data to determine where a watermark is and how to extract the watermark from stego-data.
- Type II systems require a copy of the embedded watermark for extraction and just yield a yes/no answer to the question whether the stego-data contains a watermark.

Semi-private (semi-blind) watermarking does not use the original cover-data for detection, but tries to answer the same question. (Potential application of blind and semi-blind watermarking is for evidence in court ownership,. . .)

Public (blind) watermarking – neither cover-data nor embedded watermarks are required for extraction – this is the most challenging problem.

SECRET SHARING by SECRET HIDING

A simple technique has been developed, by Naor and Shamir, that allows for a given n and $t < n$ to hide any secret (image) message m in images on transparencies in such away that each of n parties receives one transparency and

- no t - 1 parties are able to obtain the message m from the transparencies they have.
- any t of the parties can easily get (read or see) the message m just by stacking their transparencies together and aligning them carefully.

APPENDIX

SIGNAL PROCESSING TERMINOLOGY

In some applications of steganography the following signal processing technology is used.

- Payload - message to be secretly communicated;
- Carrier - data file or signal into which payload is embedded
- Package - stego file - covert message - the outcome of embedding of payload into carrier.
- Encoding density - the percentage of bytes or other signal elements into which the payload is embedded.

TO REMEMBER !!!

There is no use in trying, she said: one cannot believe impossible things.

I dare to say that you have not had much practice, said the queen,

When I was your age, I always did it for half-an-hour a day and sometimes I have believed as many as six impossible things before breakfast.

Lewis Carroll: Through the Looking-glass, 1872

Part XII

From theory to practice in cryptography

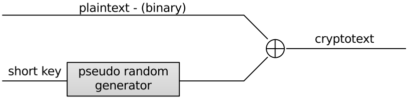
I.SHIFT REGISTERS

The first practical approach to ONE-TIME PAD cryptosystem.

I.SHIFT REGISTERS

The first practical approach to ONE-TIME PAD cryptosystem.

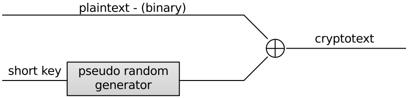
Basic idea: to use a short key, called "seed" with a pseudorandom generator to generate as long key as needed.



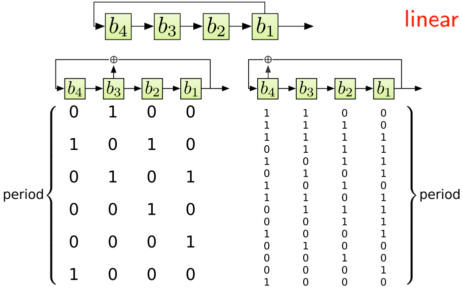
I.SHIFT REGISTERS

The first practical approach to ONE-TIME PAD cryptosystem.

Basic idea: to use a short key, called "seed" with a pseudorandom generator to generate as long key as needed.



Shift registers as pseudorandom generators



linear shift register

Theorem For every $n > 0$ there is a linear shift register of maximal period $2^n - 1$.

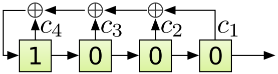
CRYPTANALYSIS of linear feedback shift registers

Sequences generated by linear shift registers have excellent statistical properties, but they are not resistant to a known plaintext attack.

Sequences generated by linear shift registers have excellent statistical properties, but they are not resistant to a known plaintext attack.

Example Let us have a 4-bit shift register and let us assume we know 8 bits of plaintext and of cryptotext. By XOR-ing these two bit sequences we get 8 bits of the output of the register (of the key), say 00011110

We need to determine c_4, c_3, c_2, c_1 such that the above sequence is outputted by the shift register



state of cell 4	state of cell 3	state of cell 2	state of cell 1
c_4	1	0	0
$c_4 \oplus c_3$	c_4	1	0
$c_2 \oplus c_4$	$c_4 \oplus c_3$	c_4	1
$c_1 \oplus c_3(c_4 \oplus c_3) \oplus c_4$	$c_2 \oplus c_4$	$c_4 \oplus c_3$	c_4

$$\begin{aligned} c_4 &= 1 \\ c_4 \oplus c_3 &= 1 \\ c_2 \oplus c_4 &= 1 \\ c_1 \oplus c_3 \oplus c_4 \oplus c_3 \cdot c_4 &= 0 \end{aligned} \quad \Rightarrow \quad \begin{aligned} c_4 &= 1 \\ c_3 &= 0 \\ c_2 &= 0 \\ c_1 &= 1 \end{aligned}$$

Linear feedback shift registers are an efficient way to realize recurrence relations of the type

$$x_{n+m} = c_0 x_n + c_1 x_{n+1} + \cdots + c_{m-1} x_{n+m-1} \text{ (mod n)}$$

that can be specified by 2m bits $c_0, \ldots, c_{m-1}$ and $x_1, \ldots, x_m$.

Linear feedback shift registers are an efficient way to realize recurrence relations of the type

$$x_{n+m} = c_0 x_n + c_1 x_{n+1} + \cdots + c_{m-1} x_{n+m-1} \pmod{n}$$

that can be specified by 2m bits $c_0, \ldots, c_{m-1}$ and $x_1, \ldots, x_m$.

Recurrences realized by shift registers on previous slides are:

$x_{n+4} = x_n$; $x_{n+4} = x_{n+2} + x_n$; $x_{n+4} = x_{n+3} + x_n$.

The main advantage of such recurrences is that a key of a very large period can be generated using a very few bits.

For example, the recurrence $x_{n+31} = x_n + x_{n+3}$, and any non-zero initial vector, produces sequences with period $2^{31} - 1$, what is more than two billions.

Linear Recurrences

Linear feedback shift registers are an efficient way to realize recurrence relations of the type

$$x_{n+m} = c_0 x_n + c_1 x_{n+1} + \cdots + c_{m-1} x_{n+m-1} \text{ (mod n)}$$

that can be specified by 2m bits $c_0, \ldots, c_{m-1}$ and $x_1, \ldots, x_m$.

Recurrences realized by shift registers on previous slides are:
$x_{n+4} = x_n$; $x_{n+4} = x_{n+2} + x_n$; $x_{n+4} = x_{n+3} + x_n$.
The main advantage of such recurrences is that a key of a very large period can be generated using a very few bits.

For example, the recurrence $x_{n+31} = x_n + x_{n+3}$, and any non-zero initial vector, produces sequences with period $2^{31} - 1$, what is more than two billions.

Encryption using one-time pad and key generated by a linear feedback shift register succumbs easily to a known plaintext attack. If we know few bits of the plaintext and of the corresponding cryptotext, one can easily determine the initial part of the key and then the corresponding linear recurrence, as already shown.

To test whether a given portion of a key was generated by a recurrence of a length m, if we know $x_1, \ldots, x_{2m}$, we need to solve the matrix equation

$$\begin{pmatrix} x_1 & x_2 & \ldots & x_m \\ x_2 & x_3 & \ldots & x_{m+1} \\ \vdots & \vdots & \ddots & \vdots \\ x_m & x_{m+1} & \ldots & x_{2m-1} \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{m-1} \end{pmatrix} = \begin{pmatrix} x_{m+1} \\ x_{m+2} \\ \vdots \\ x_{2m} \end{pmatrix}$$

and then to verify whether the remaining available bits, $x_{2m+1}, \ldots$, are really generated by the recurrence obtained.

The basic idea to find linear recurrences generating a given sequence is to check whether there is such a recurrence for m = 2, 3, ... In doing that we use the following result.

Theorem Let

$$M = \begin{pmatrix} x_1 & x_2 & \dots & x_m \\ x_2 & x_3 & \dots & x_{m+1} \\ \vdots & \vdots & \ddots & \vdots \\ x_m & x_{m+1} & \dots & x_{2m-1} \end{pmatrix}$$

If the sequence $x_1, x_2 \dots, x_{2m-1}$ satisfies a linear recurrence of length less than m, then det(M) = 0.

Conversely, if the sequence $x_1, x_2 \dots, x_{2m-1}$ satisfies a linear recurrence of length m and det(M) = 0, then the sequence also satisfies a linear recurrence of length less than m.

II. How to make cryptanalyst's task harder?

Two general methods are called diffusion and confusion.

Diffusion: dissipate the source language redundancy found in the plaintext by spreading it out over the cryptotext.

Example 1: A permutation of the plaintext rules out possibility to use frequency tables for digrams, trigrams.

Example 2: Make each letter of cryptotext to depend on so many letters of the plaintext as possible

II. How to make cryptanalyst's task harder?

Two general methods are called diffusion and confusion.

Diffusion: dissipate the source language redundancy found in the plaintext by spreading it out over the cryptotext.

Example 1: A permutation of the plaintext rules out possibility to use frequency tables for digrams, trigrams.
Example 2: Make each letter of cryptotext to depend on so many letters of the plaintext as possible

Illustration: Let letters of English be encoded by integers from $\{0, \ldots, 25\}$. Let the key $k = k_1, \ldots, k_s$ be a sequence of such integers.
Let

$$p_1, \ldots, p_n$$

be a plaintext.
Define for $0 \leq i < s$, $p_{-i} = k_{s-i}$ and construct the cryptotext by

$$c_i = \left(\sum_{j=0}^{s} p_{i-j} \right) \bmod 26,\ 1 \leq i \leq n$$

II. How to make cryptanalyst's task harder?

Two general methods are called diffusion and confusion.

Diffusion: dissipate the source language redundancy found in the plaintext by spreading it out over the cryptotext.

Example 1: A permutation of the plaintext rules out possibility to use frequency tables for digrams, trigrams.

Example 2: Make each letter of cryptotext to depend on so many letters of the plaintext as possible

Illustration: Let letters of English be encoded by integers from $\{0, \ldots, 25\}$. Let the key $k = k_1, \ldots, k_s$ be a sequence of such integers.

Let

$$p_1, \ldots, p_n$$

be a plaintext.

Define for $0 \leq i < s$, $p_{-i} = k_{s-i}$ and construct the cryptotext by

$$c_i = \left(\sum_{j=0}^{s} p_{i-j} \right) \mod 26, \; 1 \leq i \leq n$$

Confusion makes the relation between the cryptotext and plaintext as complex as possible.

Example: polyalphabetic substitutions.

Confusion and diffusion – a more detailed view

As already mentioned, two fundamental cryptographic techniques, introduced already by Shannon, are confusion and diffusion.

Confusion and diffusion – a more detailed view

As already mentioned, two fundamental cryptographic techniques, introduced already by Shannon, are confusion and diffusion.

Confusion obscures the relationship between the plaintext and the ciphertext, which makes much more difficult cryptanalyst's attempts to study cryptotext by looking for redundancies and statistical patterns. (The best way to cause confusion is through complicated substitutions.)

Confusion and diffusion – a more detailed view

As already mentioned, two fundamental cryptographic techniques, introduced already by Shannon, are confusion and diffusion.

Confusion obscures the relationship between the plaintext and the ciphertext, which makes much more difficult cryptanalyst's attempts to study cryptotext by looking for redundancies and statistical patterns. (The best way to cause confusion is through complicated substitutions.)

Diffusion dissipates redundancy of the plaintext by spreading it over cryptotext – that again makes much more difficult a cryptanalyst's attempts to search for redundancy in the plaintext through observation of cryptotext. (The best way to achieve it is through transformations that cause that bits from different positions in plaintext contribute to the same bit of cryptotext.)

Mono-alphabetic cryptosystems use no confusion and no diffusion.

Confusion and diffusion – a more detailed view

As already mentioned, two fundamental cryptographic techniques, introduced already by Shannon, are confusion and diffusion.

Confusion obscures the relationship between the plaintext and the ciphertext, which makes much more difficult cryptanalyst's attempts to study cryptotext by looking for redundancies and statistical patterns. (The best way to cause confusion is through complicated substitutions.)

Diffusion dissipates redundancy of the plaintext by spreading it over cryptotext – that again makes much more difficult a cryptanalyst's attempts to search for redundancy in the plaintext through observation of cryptotext. (The best way to achieve it is through transformations that cause that bits from different positions in plaintext contribute to the same bit of cryptotext.)

Mono-alphabetic cryptosystems use no confusion and no diffusion. Polyalphabetic cryptosystems use only confusion. In permutation cryptosystems only diffusion step is used. DES essentially uses a sequence of confusion and diffusion steps.

15. 5. 1973 National Bureau of Standards published a solicitation for a new cryptosystem.

15. 5. 1973 National Bureau of Standards published a solicitation for a new cryptosystem.

This led to the development of so far the most often used cryptosystem

Data Encryption Standard – DES

DES was developed at IBM, as a modification of an earlier cryptosystem called Lucifer.

15. 5. 1973 National Bureau of Standards published a solicitation for a new cryptosystem.

This led to the development of so far the most often used cryptosystem

Data Encryption Standard – DES

DES was developed at IBM, as a modification of an earlier cryptosystem called Lucifer.

17. 3. 1975 DES was published for the first time.

15. 5. 1973 National Bureau of Standards published a solicitation for a new cryptosystem.

This led to the development of so far the most often used cryptosystem

Data Encryption Standard – DES

DES was developed at IBM, as a modification of an earlier cryptosystem called Lucifer.

17. 3. 1975 DES was published for the first time.

After long ad heated public discussion, DES was adopted as a standard on 15. 1. 1977.

DES used to be reviewed by NBS every 5 years.

DES – description

DES was a revolutionary step in the secret-key cryptography history:

Both encryption and decryption algorithms were made public!!!!!!

DES was a revolutionary step in the secret-key cryptography history:
Both encryption and decryption algorithms were made public!!!!!!

Preprocessing: A secret 56-bit key k_{56} is chosen.
A fixed+public permutation ϕ_{56} is applied to get $\phi_{56}(k_{56})$. The first (second) part of the resulting string is taken to get a 28-bit block $C_0(D_0)$. Using a fixed+public sequence $s_1, \ldots, s_{16}$ of integers, 16 pairs of 28-bit blocks (C_i, D_i), i $=$ 1,...,16 are obtained as follows:

- $C_i(D_i)$ is obtained from $C_{i-1}(D_{i-1})$ by s_i left shifts.
- Using a fixed and public order, a 48-bit block K_i is created from each pair C_i and D_i.

DES – description

DES was a revolutionary step in the secret-key cryptography history:
Both encryption and decryption algorithms were made public!!!!!!

Preprocessing: A secret 56-bit key k_{56} is chosen.
A fixed+public permutation ϕ_{56} is applied to get $\phi_{56}(k_{56})$. The first (second) part of the resulting string is taken to get a 28-bit block $C_0(D_0)$. Using a fixed+public sequence $s_1, \ldots, s_{16}$ of integers, 16 pairs of 28-bit blocks (C_i, D_i), i = 1,...,16 are obtained as follows:

- $C_i(D_i)$ is obtained from $C_{i-1}(D_{i-1})$ by s_i left shifts.
- Using a fixed and public order, a 48-bit block K_i is created from each pair C_i and D_i.

Encryption A fixed+public permutation ϕ_{64} is applied to a 64-bits long plaintext w to get $w' = L_0 R_0$, where each of the strings L_0 and R_0 has 32 bits. 16 pairs of 32-bit blocks L_i, R_i, $1 \leq i \leq 16$, are designed using the recurrence:

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i),$$

where f is a fixed+public and easy-to-implement function.
The cryptotext $c = \phi_{64}^{-1}(L_{16}, R_{16})$

Encryption A fixed+public permutation ϕ_{64} is applied to a 64-bits long plaintext w to get $w' = L_0R_0$, where each of the strings L_0 and R_0 has 32 bits. 16 pairs of 32-bit blocks L_i, R_i, $1 \leq i \leq 16$, are designed using the recurrence:

$$\begin{gathered} L_i = R_{i-1} \\ R_i = L_{i-1} \oplus f(R_{i-1}, K_i), \end{gathered}$$

where f is a fixed+public and easy-to-implement function.
The cryptotext $c = \phi_{64}^{-1}(L_{16}, R_{16})$
Decryption $\phi_{64}(c) = L_{16}R_{16}$ is computed and then the recurrence

$$\begin{gathered} R_{i-1} = L_i \\ L_{i-1} = R_i \oplus f(L_i, K_i), \end{gathered}$$

is used to get L_i, R_i i = 15,...,1,0, w = $\phi_{64}^{-1}(L_0, R_0)$.

How fast is DES?

200 megabits can be encrypted per second using a special hardware.

How fast is DES?

200 megabits can be encrypted per second using a special hardware.

How safe is DES?

Pretty good.

How fast is DES?

200 megabits can be encrypted per second using a special hardware.

How safe is DES?

Pretty good.

How to increase security when using DES?

1. Use two keys, for a double encryption.
2. Use three keys, k_1, k_2 and k_3 to compute

$$c = DES_{k_1}(DES^{-1}_{k_2}(DES_{k_3}(w)))$$

How to increase security when encrypting long plaintexts?

$$w = m_1 m_2 \dots m_n$$

where each m_i has 64-bits.
Choose a 56-bit key k and a 64-bit block c_0 and compute

$$c_i = DES(m_i \oplus c_{i-1})$$

for i = 1,. . . ,n.

The DES controversy

1. There have been suspicions that the design of DES might contain hidden "trapdoors" what allows NSA to decrypt messages.

The DES controversy

1. There have been suspicions that the design of DES might contain hidden "trapdoors" what allows NSA to decrypt messages.
2. The main criticism has been that the size of the keyspace, 2^{56}, is too small for DES to be really secure.

The DES controversy

1. There have been suspicions that the design of DES might contain hidden "trapdoors" what allows NSA to decrypt messages.
2. The main criticism has been that the size of the keyspace, 2^{56}, is too small for DES to be really secure.
3. In 1977 Diffie+Hellamn suggested that for \$ 20 millions one could build a VLSI chip that could search the entire key space within 1 day.

The DES controversy

1. There have been suspicions that the design of DES might contain hidden "trapdoors" what allows NSA to decrypt messages.
2. The main criticism has been that the size of the keyspace, 2^{56}, is too small for DES to be really secure.
3. In 1977 Diffie+Hellamn suggested that for \$ 20 millions one could build a VLSI chip that could search the entire key space within 1 day.
4. In 1993 M. Wiener suggested a machine of the cost \$ 100.000 that could find the key in 1.5 days.

- A cryptosystem is called linear if each bit of cryptotext is a linear combination of bits of plaintext.

- A cryptosystem is called linear if each bit of cryptotext is a linear combination of bits of plaintext.
- For linear cryptosystems there is a powerful decryption method – so-called linear cryptanalysis.

What are the key elements of DES?

- A cryptosystem is called linear if each bit of cryptotext is a linear combination of bits of plaintext.
- For linear cryptosystems there is a powerful decryption method – so-called linear cryptanalysis.
- The only components of DES that are non-linear are S-boxes.
- Some of original requirements for S-boxes:
  - Each row of an S-box should include all possible output bit combinations;
  - It two inputs to an S-box differ in precisely one bit, then the output must differ in a minimum of two bits;
  - If two inputs to an S-box differ in their first two bits, but have identical last two bits, the two outputs have to be distinct.

What are the key elements of DES?

- A cryptosystem is called linear if each bit of cryptotext is a linear combination of bits of plaintext.
- For linear cryptosystems there is a powerful decryption method – so-called linear cryptanalysis.
- The only components of DES that are non-linear are S-boxes.
- Some of original requirements for S-boxes:
  - Each row of an S-box should include all possible output bit combinations;
  - It two inputs to an S-box differ in precisely one bit, then the output must differ in a minimum of two bits;
  - If two inputs to an S-box differ in their first two bits, but have identical last two bits, the two outputs have to be distinct.
- There have been many other very technical requirements for DES items in order to ensure security.

- Existence of weak keys: they are such keys k that for any plaintext p,
$$E_k(E_k(p)) = p.$$
There are four such keys:
$k \in \{(0^{28}, 0^{28}), (1^{28}, 1^{28}), (0^{28}, 1^{28}), (1^{28}, 0^{28})\}$

- Existence of weak keys: they are such keys k that for any plaintext p,
$$E_k(E_k(p)) = p.$$
There are four such keys:
$k \in \{(0^{28}, 0^{28}), (1^{28}, 1^{28}), (0^{28}, 1^{28}), (1^{28}, 0^{28})\}$
- The existence of semi-weak key pairs (k_1, k_2) such that for any plaintext
$$E_{k_1}(E_{k_2}(p)) = p.$$

- Existence of weak keys: they are such keys k that for any plaintext p,
$$E_k(E_k(p)) = p.$$
There are four such keys:
$k \in \{(0^{28}, 0^{28}), (1^{28}, 1^{28}), (0^{28}, 1^{28}), (1^{28}, 0^{28})\}$
- The existence of semi-weak key pairs (k_1, k_2) such that for any plaintext
$$E_{k_1}(E_{k_2}(p)) = p.$$
- The existence of complementation property
$$E_{c(k)}(c(p)) = c(E_k(p)),$$
where c(x) is binary complement of binary string x.

DES modes of operation

ECB mode: to encode a sequence

$$x_1, x_2, x_3, \ldots$$

of 64-bit plaintext blocks, each x_i is encrypted with the same key.

DES modes of operation

ECB mode: to encode a sequence

$$x_1, x_2, x_3, \ldots$$

of 64-bit plaintext blocks, each x_i is encrypted with the same key.

CBC mode: to encode a sequence

$$x_1, x_2, x_3, \ldots$$

of 64-bit plaintext blocks, a y_0 is chosen and each x_i is encrypted by cryptotext

$$y_i = e_k(y_{i-1} \oplus x_i).$$

DES modes of operation

ECB mode: to encode a sequence

$$x_1, x_2, x_3, \dots$$

of 64-bit plaintext blocks, each x_i is encrypted with the same key.

CBC mode: to encode a sequence

$$x_1, x_2, x_3, \dots$$

of 64-bit plaintext blocks, a y_0 is chosen and each x_i is encrypted by cryptotext

$$y_i = e_k(y_{i-1} \oplus x_i).$$

OFB mode: to encode a sequence

$$x_1, x_2, x_3, \dots$$

of 64-bit plaintext blocks, a z_0 is chosen, $z_i = e_k(z_{i-1})$ are computed and each x_i is encrypted by cryptotext $y_i = x_i \oplus z_i$.

DES modes of operation

ECB mode: to encode a sequence

$$x_1, x_2, x_3, \ldots$$

of 64-bit plaintext blocks, each x_i is encrypted with the same key.

CBC mode: to encode a sequence

$$x_1, x_2, x_3, \ldots$$

of 64-bit plaintext blocks, a y_0 is chosen and each x_i is encrypted by cryptotext

$$y_i = e_k(y_{i-1} \oplus x_i).$$

OFB mode: to encode a sequence

$$x_1, x_2, x_3, \ldots$$

of 64-bit plaintext blocks, a z_0 is chosen, $z_i = e_k(z_{i-1})$ are computed and each x_i is encrypted by cryptotext $y_i = x_i \oplus z_i$.

CFB mode: to encode a sequence

$$x_1, x_2, x_3, \ldots$$

of 64-bit plaintext blocks a y_0 is chosen and each x_i is encrypted by cryptotext

$$y_i = x_i \oplus z, \text{ where } z_i = e_k(y_{i-1}).$$

8-bit VERSION of the CFB MODE

In this mode each 8-bit piece of the plaintext is encrypted without having to wait for an entire block to be available.

The plaintext is broken into 8-bit pieces: P=$[P_1, P_2, \ldots]$.

Encryption: An initial 64-bit block X_1 is chosen and then, for j=1,2,. . . , the following computation is done:

$$C_j = P_j \oplus L_8(e_k(X_j))$$
$$X_{j+1} = R_{56}(X_j) \| C_j,$$

8-bit VERSION of the CFB MODE

In this mode each 8-bit piece of the plaintext is encrypted without having to wait for an entire block to be available.

The plaintext is broken into 8-bit pieces: P=$[P_1, P_2, \ldots]$.

Encryption: An initial 64-bit block X_1 is chosen and then, for j=1,2,. . . , the following computation is done:

$$C_j = P_j \oplus L_8(e_k(X_j))$$
$$X_{j+1} = R_{56}(X_j) \| C_j,$$

$L_8(X)$ denotes the 8 leftmost bits of X. $R_{56}(X)$ denotes the rightmost 56 bits of X. $X \| Y$ denotes concatenation of strings X and Y.

Decryption:

$$P_j = C_j \oplus L_8(e_k(X_j))$$
$$X_{j+1} = R_{56}(X_j) \| C_j,$$

- CBC mode is used for block-encryption and also for authentication;
- CFB mode is used for stream-encryption;
- OFB mode is used for stream-encryptions that require message authentication;

CTR MODE

Counter Mode – some consider it as the best one.
Key design: $k_i = E_k(n, i)$ for a nonce n;
Encryption: $y_i = x_i \oplus k_i$
This mode is very fast because a key stream can be parallelised to any degree. Because of that this mode is used in network security applications.

- In 1993 M. J. Weiner suggested that one could design, using one million dollars, a computer capable to decrypt, using brute force, DES in 3.5 hours.

Killers and death of DES

- In 1993 M. J. Weiner suggested that one could design, using one million dollars, a computer capable to decrypt, using brute force, DES in 3.5 hours.
- In 1998 group of P. Kocher designed, using a quarter million of dolars, a computer capable to decrypt DES in 56 hours.

- In 1993 M. J. Weiner suggested that one could design, using one million dollars, a computer capable to decrypt, using brute force, DES in 3.5 hours.
- In 1998 group of P. Kocher designed, using a quarter million of dolars, a computer capable to decrypt DES in 56 hours.
- In 1999 they did that in 24 hours.

- In 1993 M. J. Weiner suggested that one could design, using one million dollars, a computer capable to decrypt, using brute force, DES in 3.5 hours.
- In 1998 group of P. Kocher designed, using a quarter million of dolars, a computer capable to decrypt DES in 56 hours.
- In 1999 they did that in 24 hours.
- It started to be clear that a new cryptosystem with larger keys is badly needed.

Design of several important practical cryptosystems used the following three general design principles for cryptosystems.

Design of several important practical cryptosystems used the following three general design principles for cryptosystems.

A product cryptosystem combines two or more crypto-transformations in such a way that resulting cryptosystem is more secure than component transformations.

Design of several important practical cryptosystems used the following three general design principles for cryptosystems.

A product cryptosystem combines two or more crypto-transformations in such a way that resulting cryptosystem is more secure than component transformations.

An iterated block cryptosystem iteratively uses a round function (and it has as parameters number of rounds r, block bit-size n, subkeys bit-size k) of the input key K from which r subkeys K_i are derived.

Design of several important practical cryptosystems used the following three general design principles for cryptosystems.

A product cryptosystem combines two or more crypto-transformations in such a way that resulting cryptosystem is more secure than component transformations.

An iterated block cryptosystem iteratively uses a round function (and it has as parameters number of rounds r, block bit-size n, subkeys bit-size k) of the input key K from which r subkeys K_i are derived.

A Feistel cryptosystem is an iterated cryptosystem mapping 2t-bit plaintext (L_0, R_0) of t-bit blocks L_0 and R_0 to a 2t-bit cryptotext (R_r, L_r), through an r-round process, where $r > 0$.

For $0 < I < r + 1$, the round i maps (L_{i-1}, R_{i-1}) to (L_i, R_i) using a subkey K_i as follows

$$L_i = R_{i-1},\ R_i = K_{i-1} \oplus f(R_{i-1}, K_i),$$

where each subkey K_i is derived from the main key K.

- Blowfish is Feistel type cryptosystem developed in 1994 by Bruce Schneier.
- Blowfish is more secure and faster than DES.
- It encrypts 8-bytes blocks into 8-bytes blocks.
- Key length is variable 32k, for $k = 1, 2, \ldots, 16$.
- For decryption it does not reverse the order of encryption, but it follows it.
- S-boxes are key dependent and they, as well as subkeys are created by repeated execution of Blowfish enciphering transformation.
- Blowfish has very strong avalanche effect.
- A follower of Blowfish, Twofish, was one of 5 candidates for AES.
- Blowfish can be downloaded free from the B. Schneier web site.

AES CRYPTOSYSTEM

On October 2, 2000, NIST selected, as new **A**dvanced **E**ncryption **S**tandard, the cryptosystem Rijndael, designed in 1998 by Joan Daemen and Vincent Rijmen.

AES CRYPTOSYSTEM

On October 2, 2000, NIST selected, as new **A**dvanced **E**ncryption **S**tandard, the cryptosystem Rijndael, designed in 1998 by Joan Daemen and Vincent Rijmen.

The main goal has been to develop a new cryptographic standard that could be used to encrypt sensitive governmental information securely, well into the next century.

AES CRYPTOSYSTEM

On October 2, 2000, NIST selected, as new **A**dvanced **E**ncryption **S**tandard, the cryptosystem Rijndael, designed in 1998 by Joan Daemen and Vincent Rijmen.

The main goal has been to develop a new cryptographic standard that could be used to encrypt sensitive governmental information securely, well into the next century.

AES was expected to be used obligatory by U.S. governmental institution and, naturally, voluntarily, but as a necessity, also by the private sector.

AES CRYPTOSYSTEM

On October 2, 2000, NIST selected, as new **A**dvanced **E**ncryption **S**tandard, the cryptosystem Rijndael, designed in 1998 by Joan Daemen and Vincent Rijmen.

The main goal has been to develop a new cryptographic standard that could be used to encrypt sensitive governmental information securely, well into the next century.

AES was expected to be used obligatory by U.S. governmental institution and, naturally, voluntarily, but as a necessity, also by the private sector.

AES is to encrypt 128-bit blocks using a key with 128, 192 or 256 bits. In addition, AES is to be used as a standard for authentication (MAC), hashing and pseudorandom numbers generation.

AES CRYPTOSYSTEM

On October 2, 2000, NIST selected, as new **A**dvanced **E**ncryption **S**tandard, the cryptosystem Rijndael, designed in 1998 by Joan Daemen and Vincent Rijmen.

The main goal has been to develop a new cryptographic standard that could be used to encrypt sensitive governmental information securely, well into the next century.

AES was expected to be used obligatory by U.S. governmental institution and, naturally, voluntarily, but as a necessity, also by the private sector.

AES is to encrypt 128-bit blocks using a key with 128, 192 or 256 bits. In addition, AES is to be used as a standard for authentication (MAC), hashing and pseudorandom numbers generation.

Motivations and advantages of AES:

- Short code and fast implementations
- Simplicity and transparency of the design
- Variable key length
- Resistance against all known attacks

ARITHMETIC in GF(2^8)

The basic data structure of AES is a byte

$$a = (a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0)$$

where a_i's are bits, which can be conveniently represented by the polynomial

$$a(x) = a_7x^7 + a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0.$$

Bytes can be conveniently seen as elements of the field

$$F = GF(2^8)/m(x), \text{ where } m(x) = x^8 + x^4 + x^3 + x + 1.$$

In the field F, the addition is the bit-wise-XOR and multiplication can be elegantly expressed using polynomial multiplication modulo m(x).

$$c = a \oplus b; \; c = a \bullet b \text{ where } c(x) = [a(x) \bullet b(x)] \mod m(x)$$

MULTIPLICATION in GF(2^8)

Multiplication

$$c = a \bullet b \text{ where } c(x) = [a(x) \bullet b(x)] \mod m(x)$$

in GF(2^8) can be easily performed using a new operation

$$\text{b} = \text{xtime(a)}$$

that corresponds to the polynomial multiplication

$$b(x) = [a(x) \bullet x] \mod m(x),$$

as follows

set c = 00000000 and p = a;

for i = 0 **to** 7 **do**

$c \leftarrow c \oplus (b_i \bullet p)$

$p \leftarrow xtime(p)$

Hardware implementation of the multiplication requires therefore one circuit for operation xtime and two 8-bit registers.

Operation b = xtime(a) can be implemented by one step (shift) of the following shift register:

EXAMPLES

- $$\text{'53'} + \text{'87'} = \text{'D4'}$$

  because, in binary,

  $$\text{'01010011'} \oplus \text{'10000111'} = \text{'11010100'}$$

  what means

  $$(x^6 + x^4 + x + 1) + (x^7 + x^2 + x + 1) = x^7 + x^6 + x^4 + x^2$$

EXAMPLES

- $$\text{‘53‘} + \text{‘87’} = \text{‘D4‘}$$

  because, in binary,

  $$\text{‘01010011‘} \oplus \text{‘10000111‘} = \text{‘11010100’}$$

  what means

  $$(x^6 + x^4 + x + 1) + (x^7 + x^2 + x + 1) = x^7 + x^6 + x^4 + x^2$$

- $$\text{‘57’‘} \bullet \text{‘83‘} = \text{‘C1’}$$

  Indeed,

  $$(x^6 + x^4 + x^2 + x + 1)(x^7 + x + 1) = x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$$

  and

  $$(x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1) \bmod (x^8 + x^4 + x^3 + x + 1) = x^7 + x^6 + 1$$

EXAMPLES

- $$\text{‘53‘} + \text{‘87’} = \text{‘D4‘}$$

  because, in binary,

  $$\text{‘01010011‘} \oplus \text{‘10000111‘} = \text{‘11010100‘}$$

  what means

  $$(x^6 + x^4 + x + 1) + (x^7 + x^2 + x + 1) = x^7 + x^6 + x^4 + x^2$$

- $$\text{‘57’‘} \bullet \text{‘83‘} = \text{‘C1’}$$

  Indeed,

  $$(x^6 + x^4 + x^2 + x + 1)(x^7 + x + 1) = x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$$

  and

  $$(x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1) \\ \bmod (x^8 + x^4 + x^3 + x + 1) = x^7 + x^6 + 1$$

- $$\text{‘57‘} \bullet \text{‘13‘} = (\text{‘57‘} \bullet \text{‘01’}) \oplus (\text{‘57‘} \bullet \text{‘02’}) \oplus (\text{‘57‘} \bullet \text{‘10’}) = \text{‘57‘} \oplus \text{‘AE‘} \oplus \text{‘07‘} = \text{‘FE‘}$$

  because

  $$\begin{aligned} \text{‘57‘} \bullet \text{‘02‘} &= \text{xtime}(57) = \text{‘AE‘} \\ \text{‘57‘} \bullet \text{‘04‘} &= \text{xtime}(AE) = \text{‘47‘} \\ \text{‘57‘} \bullet \text{‘08‘} &= \text{xtime}(47) = \text{‘8E‘} \\ \text{‘57‘} \bullet \text{‘10‘} &= \text{xtime}(8E) = \text{‘07’} \end{aligned}$$

Algorithms of AES work with 4-byte vectors that can be represented by polynomials of the degree at most 4 with coefficients in GF(2^8).

Addition of such polynomials is done using component-wise and bit-wise XOR.
Multiplication is done modulo $M(x) = x^4 + 1$. (It holds $x^J \mod (x^4 + 1) = x^{J \mod 4}$.)

Multiplication of vectors

$$(a_3x^3 + a_2x^2 + a_1x + a_0) \otimes (b_3x^3 + b_2x^2 + b_1x + b_0)$$

can be done using matrix multiplication

$$\begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{pmatrix} = \begin{pmatrix} a_0 & a_1 & a_2 & a_3 \\ a_1 & a_2 & a_3 & a_0 \\ a_2 & a_3 & a_0 & a_1 \\ a_3 & a_0 & a_1 & a_2 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix},$$

where additions and multiplications $(\cdot)$ are done in GF(2^8) as described before.
Multiplication of a polynomial a(x) by x results in a cyclic shift of the coefficients.

BYTE SUBSTITUTION

Byte substitution $b = \text{SubByte}(a)$ is defined by the following matrix operations

$$\begin{pmatrix} b_7 \\ b_6 \\ b_5 \\ b_4 \\ b_3 \\ b_2 \\ b_1 \\ b_0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} (a^{-1})_7 \\ (a^{-1})_6 \\ (a^{-1})_5 \\ (a^{-1})_4 \\ (a^{-1})_3 \\ (a^{-1})_2 \\ (a^{-1})_1 \\ (a^{-1})_0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

This operation is computationally heavy and it is assumed that it will be implemented by a pre-computed substitution table.

ENCRYPTION in AES

Encryption and decryption are done using state matrices

A	E	I	M
B	F	J	N
C	G	K	O
D	H	L	P

elements of which are bytes.

A byte-matrix with 4 rows and k = 4, 6 or 8 columns is also used to write down a key with $D_k = 128$, 192 or 256 bits.

Encryption and decryption are done using state matrices

A	E	I	M
B	F	J	N
C	G	K	O
D	H	L	P

elements of which are bytes.

A byte-matrix with 4 rows and k = 4, 6 or 8 columns is also used to write down a key with $D_k = 128$, 192 or 256 bits.

ENCRYPTION ALGORITHM

1. KeyExpansion

ENCRYPTION in AES

Encryption and decryption are done using state matrices

A	E	I	M
B	F	J	N
C	G	K	O
D	H	L	P

elements of which are bytes.

A byte-matrix with 4 rows and k = 4, 6 or 8 columns is also used to write down a key with $D_k = 128$, 192 or 256 bits.

ENCRYPTION ALGORITHM

1. KeyExpansion
2. AddRoundKey

Encryption and decryption are done using state matrices

A	E	I	M
B	F	J	N
C	G	K	O
D	H	L	P

elements of which are bytes.

A byte-matrix with 4 rows and k = 4, 6 or 8 columns is also used to write down a key with $D_k = 128$, 192 or 256 bits.

ENCRYPTION ALGORITHM

1. KeyExpansion
2. AddRoundKey
3. **do** (k + 5)-times:
   a) SubByte
   b) ShiftRow
   c) MixColumn
   d) AddRoundKey

Encryption and decryption are done using state matrices

A	E	I	M
B	F	J	N
C	G	K	O
D	H	L	P

elements of which are bytes.

A byte-matrix with 4 rows and k = 4, 6 or 8 columns is also used to write down a key with D_k = 128, 192 or 256 bits.

ENCRYPTION ALGORITHM

1. KeyExpansion
2. AddRoundKey
3. **do** (k + 5)-times:
   a) SubByte
   b) ShiftRow
   c) MixColumn
   d) AddRoundKey
4. Final round
   a) SubByte
   b) ShiftRow
   c) AddRoundKey

ENCRYPTION in AES

Encryption and decryption are done using state matrices

A	E	I	M
B	F	J	N
C	G	K	O
D	H	L	P

elements of which are bytes.

A byte-matrix with 4 rows and k = 4, 6 or 8 columns is also used to write down a key with $D_k = 128$, 192 or 256 bits.

ENCRYPTION ALGORITHM

1. KeyExpansion
2. AddRoundKey
3. **do** (k + 5)-times:
   a) SubByte
   b) ShiftRow
   c) MixColumn
   d) AddRoundKey
4. Final round
   a) SubByte
   b) ShiftRow
   c) AddRoundKey

The final round does not contain MixColumn procedure. The reason being is to be able to use the same hardware for encryption and decryption.

The basic key is written into the state matrix with 4, 6 or 8 columns. The goal of the key expansion procedure is to extend the number of keys in such a way that each time a key is used actually a new key is used.

The key extension algorithm generates new columns W_i of the state matrix from the columns W_{i-1} and W_{i-k} using the following rule

$$W_i = W_{i-k} \oplus V,$$

where

$$V = \begin{cases} F(W_{i-1}), & \text{if i mod k} = 0 \\ G(W_{i-1}), & \text{if i mod k} = 4 \text{ and } D_k = 256 \text{ bits}, \\ W_{i-1} & \text{otherwise} \end{cases}$$

where the function G performs only the byte-substitution of the corresponding bytes. **F**unction F is defined in a quite a complicated way.

AddRoundKey procedure adds byte-wise and bit-wise current key to the current contents of the state matrix.

ShiftRow procedure cyclically shifts i-th row of the state matrix by i shifts.

MixColumns procedure multiplies columns of the state matrix by the matrix

$$\begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix}$$

Steps of the encryption algorithm map an input state matrix into an output matrix. All encryption operations have inverse operations. Decryption algorithm applies, in the opposite order as at the encryption, the inverse versions of the encryption operations.

DECRYPTION

1. Key Expansion

DECRYPTION in AES

Steps of the encryption algorithm map an input state matrix into an output matrix. All encryption operations have inverse operations. Decryption algorithm applies, in the opposite order as at the encryption, the inverse versions of the encryption operations.

DECRYPTION

1. Key Expansion
2. AddRoundKey

Steps of the encryption algorithm map an input state matrix into an output matrix. All encryption operations have inverse operations. Decryption algorithm applies, in the opposite order as at the encryption, the inverse versions of the encryption operations.

DECRYPTION

1. Key Expansion
2. AddRoundKey
3. **do** k+5 - times:
   a) InvSubByte
   b) InvShiftRow
   c) InvMixColumn
   d) AddInvRoundKey

Steps of the encryption algorithm map an input state matrix into an output matrix. All encryption operations have inverse operations. Decryption algorithm applies, in the opposite order as at the encryption, the inverse versions of the encryption operations.

DECRYPTION

1. Key Expansion
2. AddRoundKey
3. **do** k+5 - times:
   a) InvSubByte
   b) InvShiftRow
   c) InvMixColumn
   d) AddInvRoundKey
4. Final round
   a) InvSubByte
   b) InvShiftRow
   c) AddInvRoundKey

SECURITY GOALS

The goal of the authors was that Rijndael (AES) is K-secure and hermetic in the following sense:

Definition A cryptosystem is K-secure if all possible attack strategies for it have the same expected work factor and storage requirements as for the majority of possible cryptosystems with the same security.

Definition A block cryptosystem is hermetic if it does not have weaknesses that are not present for the majority of cryptosystems with the same block and key length.

Pronunciation of the name **Rijndael** is as “Reign Dahl'‘ or “rain Doll” or “Rhine Dahl”.

PKC versus SKC – comparisons

Security: If PKC is used, only one party needs to keep secret a (single) key; If SKC is used, both party needs to keep secret one key. No PKC has been shown perfectly secure. Perfect secrecy has been shown for One-time Pad and for quantum generation of classical keys.

Longevity: With PKC, keys may need to be kept secure for (very) long time; with SKC a change of keys for each session is recommended.

Key management: If a multiuser network is used, then fewer private keys are required with PKC than with SKC.

Key exchange: With PKC no key exchange between communicating parties is needed; with SKC a hard-to-implement secret key exchange is needed.

Digital signatures: Only PKC are usable for digital signatures.

Efficiency: PKC is much slower than SKC (10 times when software implementations of RSA and DES are compared).

Key sizes: Keys for PKC (2048 bits for RSA) are significantly larger than for SCK (128 bits for AES).

Non-repudiation: With PKC we can ensure, using digital signatures, non-repudiation, but not with SKC.

Modern cryptography uses both SKC and PKC, in so-called **hybrid cryptosystems** or in **digital envelopes** to send a message m using a secret key k, public encryption exponent e, and secret decryption exponent d, as follows:

1. Key k is encrypted using e and sent as e(k)
2. Secret description exponent d is used to get k=d(e(k))
3. SKC with k is then used to encrypt a message

KEY MANAGEMENT

Secure methods of key management are extremely important. In practice, most of the attacks on public-key cryptosystems are likely to be at the key management levels.

Problems: How to obtain securely an appropriate key pair? How to get other people's public keys? How to get confidence in the legitimacy of other's public keys? How to store keys? How to set, extend,... expiration dates of the keys?

KEY MANAGEMENT

Secure methods of key management are extremely important. In practice, most of the attacks on public-key cryptosystems are likely to be at the key management levels.

Problems: How to obtain securely an appropriate key pair? How to get other people's public keys? How to get confidence in the legitimacy of other's public keys? How to store keys? How to set, extend,... expiration dates of the keys?

Who needs a key? Anyone wishing to sign a message, to verify signatures, to encrypt messages and to decrypt messages.

How does one get a key pair? Each user should generate his/her own key pair. Once generated, a user must register his/her public-key with some central administration, called a certifying authority. This authority returns a certificate.

Certificates are digital documents attesting to the binding of a public-key to an individual or institutions. They allow verification of the claim that a given public-key does belong to a given individual. Certificates help to prevent someone from using a phony key to impersonate someone else. In their simplest form, certificates contain a public-key and a name. In addition they contain: expiration date, name of the certificate issuing authority, serial number of the certificate and the digital signature of the certificate issuer.

How are certificates used – certification authorities

The most secure use of authentication involves enclosing one or more certificates with every signed message. The receiver of the message verifies the certificate using the certifying authorities public-keys and, being confident of the public-keys of the sender, verifies the message's signature. There may be more certificates enclosed with a message, forming a hierarchical chain, wherein one certificate testifies to the authenticity of the previous certificate. At the top end of a certificate hierarchy is a top-level certifying-authority to be trusted without a certificate.

Example According to the standards, every signature points to a certificate that validates the public-key of the signer. Specifically, each signature contains the name of the issuer of the certificate and the serial number of the certificate.

The most secure use of authentication involves enclosing one or more certificates with every signed message. The receiver of the message verifies the certificate using the certifying authorities public-keys and, being confident of the public-keys of the sender, verifies the message's signature. There may be more certificates enclosed with a message, forming a hierarchical chain, wherein one certificate testifies to the authenticity of the previous certificate. At the top end of a certificate hierarchy is a top-level certifying-authority to be trusted without a certificate.

Example According to the standards, every signature points to a certificate that validates the public-key of the signer. Specifically, each signature contains the name of the issuer of the certificate and the serial number of the certificate.

How do certifying authorities store their private keys?

It is extremely important that private-keys of certifying authorities are stored securely. One method to store the key in a tamper-proof box called a Certificate Signing Unit, CSU.

The CSU should, preferably, destroy its contents if ever opened. Not even employees of the certifying authority should have access to the private-key itself, but only the ability to use private-key in the certificates issuing process.

CSU are for sells

Note: PKCS – Public Key Certification Standards.

What is PKI?

- PKI (Public Key Infrastructure) is an infrastructure that allows to handle public-key problems for the community that uses public-key cryptography.

- PKI (Public Key Infrastructure) is an infrastructure that allows to handle public-key problems for the community that uses public-key cryptography.
- Structure of PKI

  Security policy that specifies rules under which PKI can be handled.

  Products that generate, store, distribute and manipulate keys.

  Procedures that define methods
  - to generate and manipulate keys
  - to generate and manipulate certificates
  - to distribute keys and certificates
  - to use certificates.
- Authorities that take care that the general security policy is fully performed.

PKI users and systems

- Certificate holder
- Certificate user
- Certification authority (CA)
- Registration authority (RA)
- Revocation authority
- Repository (to publish a list of certificates, of relocated certificates,...)
- Policy management authority (to create certification policy)
- Policy approving authority

SECURITY of Certification and Registration authorities

PKI system is so secure how secure are systems for certificate authorities (CA) and registration authorities (RA).

Basic principles to follow to ensure necessary security of CA and RA.

- Private key of CA has to be stored in a way that is secure against intentional professional attacks.
- Steps have to be made for renovation of the private key in the case of a collapse of the system.
- Access to CA/RA tools has to be maximally controlled.
- Each requirement for certification has to be authorized by several independent operators.
- All key transactions of CA/RA have to be logged to be available for a possible audit.
- All CA/RA systems and their documentation have to satisfy maximal requirements for their reliability.

Public-key cryptography has low infrastructure overhead, it is more secure, more truthful and with better geographical reach. However, this is due to the fact that public-key users bear a substantial administrative burden and security advantages of the public key cryptography rely excessively on the end-users' security discipline.

Problem 1: With public-key cryptography users must constantly be careful to validate rigorously every public-key they use and must take care for secrecy of their private secret keys.

PUBLIC-KEY INFRASTRUCTURE PROBLEMS

Public-key cryptography has low infrastructure overhead, it is more secure, more truthful and with better geographical reach. However, this is due to the fact that public-key users bear a substantial administrative burden and security advantages of the public key cryptography rely excessively on the end-users' security discipline.

Problem 1: With public-key cryptography users must constantly be careful to validate rigorously every public-key they use and must take care for secrecy of their private secret keys.

Problem 2: End-users are rarely willing or able to manage keys sufficiently carefully.

User's behavior is the weak link in any security system, and public-key security is unable to reinforce this weakness.

PUBLIC-KEY INFRASTRUCTURE PROBLEMS

Public-key cryptography has low infrastructure overhead, it is more secure, more truthful and with better geographical reach. However, this is due to the fact that public-key users bear a substantial administrative burden and security advantages of the public key cryptography rely excessively on the end-users' security discipline.

Problem 1: With public-key cryptography users must constantly be careful to validate rigorously every public-key they use and must take care for secrecy of their private secret keys.

Problem 2: End-users are rarely willing or able to manage keys sufficiently carefully.

User's behavior is the weak link in any security system, and public-key security is unable to reinforce this weakness.

Problem 3: Only sophisticated users, like system administrators, can realistically be expected to meet fully the demands of public-key cryptography.

Main components of public-key infrastructure

- The Certification Authority (CA) signs user's public-keys.
  (There has to be a hierarchy of CA, with a root CA on the top.)
- The Directory is a public-access database of valid certificates.
- The Certificate Revocation List (CRL) – a public-access database of invalid certificates. (There has to be a hierarchy of CRL).

Main components of public-key infrastructure

- The Certification Authority (CA) signs user's public-keys.
  (There has to be a hierarchy of CA, with a root CA on the top.)
- The Directory is a public-access database of valid certificates.
- The Certificate Revocation List (CRL) – a public-access database of invalid certificates. (There has to be a hierarchy of CRL).

Stages at which key management issues arise

- Key creation: user creates a new key pair, proves his identify to CA. CA signs a certificate. User encrypts his private key.
- Single sign-on: decryption of the private key, participation in public-key protocols.
- Key revocation: CRL should be checked every time a certificate is used. If a user's secret key is compromised, CRL administration has to be notified.

MAIN PROBLEMS

- Authenticating the users: How does a CA authenticate a distant user, when issuing the initial certificate?
  (Ideally CA and the user should meet. Consequently, properly authenticated certificates will have to be expensive, due to the label cost in a face-to-face identity check.)
- Authenticating the CA: Public key cryptography cannot secure the distribution and the validation of the Root CA's public key.
- Certificate revocation lists: Timely and secure revocation presents big scaling and performance problems. As a result public-key deployment is usually proceeding without a revocation infrastructure.
  (Revocation is the classical Achilles' Heel of public-key cryptography.)
- Private key management: The user must keep his long-lived secret key in memory during his login-session: There is no way to force a public-key user to choose a good password.
  (Lacking effective password-quality controls, most public-key systems are vulnerable to the off-line guessing attacks.)

LIFE CYCLE of CERTIFICATES

Issuing of certificates

- registration of applicants for certificates;
- generation of pairs of keys;
- creation of certificates;
- delivering of certificates;
- dissemination of certificates;
- backuping of keys;

LIFE CYCLE of CERTIFICATES

Issuing of certificates

- registration of applicants for certificates;
- generation of pairs of keys;
- creation of certificates;
- delivering of certificates;
- dissemination of certificates;
- backuping of keys;

Using of certificates

- receiving a certificate;
- validation of the certificate;
- key backup and recovery;
- automatic key/certificate updating

LIFE CYCLE of CERTIFICATES

Issuing of certificates

- registration of applicants for certificates;
- generation of pairs of keys;
- creation of certificates;
- delivering of certificates;
- dissemination of certificates;
- backuping of keys;

Using of certificates

- receiving a certificate;
- validation of the certificate;
- key backup and recovery;
- automatic key/certificate updating

Revocation of certificates

- expiration of certificates validity period;
- revocation of certificates;
- archivation of keys and certificates.

In June 1991 Phil Zimmermann, made publicly available software that made use of RSA cryptosystem very friendly and easy and by that he made strong cryptography widely available.

Starting February 1993 Zimmermann was for three years a subject of FBI and Grand Jury investigations, being accused of illegal exporting arms (strong cryptography tools).

William Cowell, Deputy Director of NSA said: "If all personal computers in the world - approximately 200 millions – were to be put to work on a single PGP encrypted message, it would take an average an estimated 12 million times the age of universe to break a single message".

Heated discussion whether strong cryptography should be allowed keep going on. September 11 attack brought another dimension into the problem.

SECURITY / PRIVACY REALITY and TOOLS

Concerning security we are winning battles, but we are loosing wars concerning privacy.

Four areas concerning security and privacy:

- Security of communications – cryptography
- Computer security (operating systems, viruses, . . .)
- Physical security
- Identification and biometrics

With Google we lost privacy.

Techniques that are indeed used to break cryptosystems:

By NSA:

- By exhaustive search (up to 2^{80} options).
- By exploiting specific mathematical and statistical weaknesses to speed up the exhaustive search.
- By selling compromised crypto-devices.
- By analysing crypto-operators methods and customs.

By FBI:

- Using keystroke analysis.
- Using the fact that in practice long keys are almost always designed from short guessable passwords.

APPENDIX

- 660-bits integers were already (factorized) broken in practice.
- 1024-bits integers are currently used as moduli.
- 512-bit integers can be factorized with a device costing 5 K $ in about 10 minutes.
- 1024-bit integers could be factorized in 6 weeks by a device costing 10 millions of dollars.

Patentability of cryptography

- Cryptographic systems are patentable
- Many secret-key cryptosystems have been patented
- The basic idea of public-key cryptography are contained in U.S. Patents 4 200 770 (M. Hellman, W. Diffie, R. Merkle) – 29. 4. 1980 U.S. Patent 4 218 582 (M. Hellman, R. Merkle)

The exclusive licensing rights to both patents are held by "Public Key Partners" (PKP) which also holds rights to the RSA patent.

All legal challenges to public-key patents have been so far settled before judgment.

Some patent applications for cryptosystems have been blocked by intervention of US: intelligence or defense agencies.

All cryptographic products in USA needed export licences from the State department, acting under authority of the International Traffic in Arms Regulation, which defines cryptographic devices, including software, as munition.

Export of cryptography for authentication has not been restricted, Problems were only whith cryptography for privacy.

Part XIII

Quantum cryptography

Quantum cryptography has a potential to be cryptography of 21^{st} century.

An important new feature of quantum cryptography is that security of quantum cryptographic protocols is based on the laws of nature – of quantum physics, and not on the unproven assumptions of computational complexity.

Quantum cryptography is the first area of information processing and communication in which quantum particle physics laws are directly exploited to bring an essential advantage in information processing.

- It has been shown that would we have quantum computer, we could design absolutely secure quantum generation of shared and secret random classical keys.
- It has been proven that even without quantum computers unconditionally secure quantum generation of classical secret and shared keys is possible (in the sense that any eavesdropping is detectable).
- Unconditionally secure basic quantum cryptographic primitives, such as bit commitment and oblivious transfer, are impossible.
- Quantum zero-knowledge proofs exist for all NP-complete languages
- Quantum teleportation and pseudo-telepathy are possible.
- Quantum cryptography and quantum networks are already in advanced experimental stage.

As an introduction to quantum cryptography

the very basic motivations, experiments, principles, concepts and results of quantum information processing and communication

will be presented in the next few slides.

BASIC MOTIVATION

In quantum information processing we witness an interaction between the two most important areas of science and technology of 20-th century, between

quantum physics and informatics.

This is very likely to have important consequences for 21th century.

QUANTUM PHYSICS

Quantum physics deals with fundamental entities of physics – **particles** (waves?) like

- protons, electrons and neutrons (from which matter is built);
- photons (which carry electromagnetic radiation)
- various "elementary particles" which mediate other interactions in physics.
- We call them particles in spite of the fact that some of their properties are totally unlike the properties of what we call particles in our ordinary classical world.

  For example, a quantum particle can go through two places at the same time and can interact with itself.

  Because of that quantum physics is full of counter-intuitive, weird, mysterious and even paradoxical events.

FEYNMAN's VIEW

I am going to tell you what Nature behaves like ...

However, do not keep saying to yourself, if you can possibly avoid it,

BUT HOW CAN IT BE LIKE THAT?

Because you will get "down the drain" into a blind alley from which nobody has yet escaped

NOBODY KNOWS HOW IT CAN BE LIKE THAT

Richard Feynman (1965): The character of physical law.

CLASSICAL versus QUANTUM INFORMATION

Main properties of classical information:

1. It is easy to store, transmit and process classical information in time and space.
2. It is easy to make (unlimited number of) copies of classical information
3. One can measure classical information without disturbing it.

Main properties of quantum information:

1. It is difficult to store, transmit and process quantum information
2. There is no way to copy unknown quantum information
3. Measurement of quantum information destroys it, in general.

The essence of the difference between
classical computers and **quantum computers**
is in the way information is stored and processed.

In classical computers, information is represented on macroscopic level by bits, which can take one of the two values

0 or 1

In quantum computers, information is represented on microscopic level using qubits, (quantum bits) which can take on any from the following uncountable many values

$$\alpha|0\rangle + \beta|1\rangle$$

where α, β are arbitrary complex numbers such that

$$|\alpha|^2 + |\beta|^2 = 1.$$

CLASSICAL versus QUANTUM REGISTERS

An n bit classical register can store at any moment exactly one n-bit string.

An n-qubit quantum register can store at any moment a superposition of all 2^n n-bit strings.

Consequently, on a quantum computer one can compute in a single step with 2^n values.

This enormous massive parallelism is one reason why quantum computing can be so powerful.

CLASSICAL EXPERIMENTS

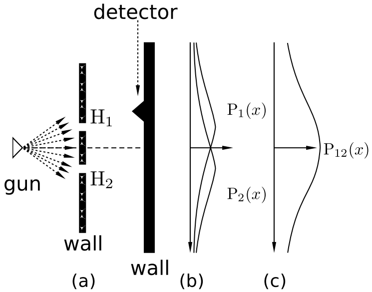


Figure 1: Experiment with bullets

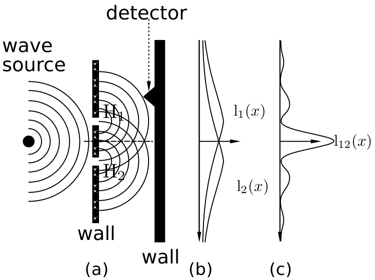


Figure 2: Experiments with waves

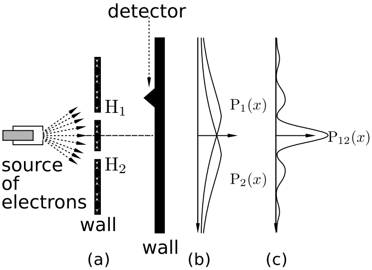


Figure 3: Two-slit experiment

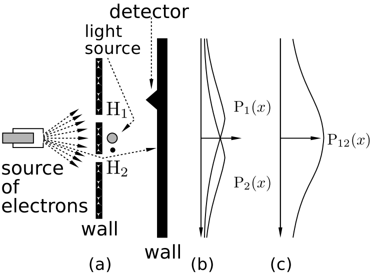


Figure 4: Two-slit experiment with an observation

THREE BASIC PRINCIPLES

P1 To each transfer from a quantum state ϕ to a state ψ a complex number

$$\langle\psi|\phi\rangle$$

is associated. This number is called the probability amplitude of the transfer and

$$|\langle\psi|\phi\rangle|^2$$

is then the **probability** of the transfer.

THREE BASIC PRINCIPLES

P1 To each transfer from a quantum state ϕ to a state ψ a complex number

$$\langle\psi|\phi\rangle$$

is associated. This number is called the probability amplitude of the transfer and

$$|\langle\psi|\phi\rangle|^2$$

is then the **probability** of the transfer.

P2 If a transfer from a quantum state ϕ to a quantum state ψ can be decomposed into two subsequent transfers

$$\psi \leftarrow \phi' \leftarrow \phi$$

then the resulting amplitude of the transfer is the product of amplitudes of subtransfers: $\langle\psi|\phi\rangle = \langle\psi|\phi'\rangle\langle\phi'|\phi\rangle$

THREE BASIC PRINCIPLES

P1 To each transfer from a quantum state ϕ to a state ψ a complex number

$$\langle\psi|\phi\rangle$$

is associated. This number is called the probability amplitude of the transfer and

$$|\langle\psi|\phi\rangle|^2$$

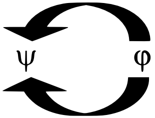
is then the **probability** of the transfer.

P2 If a transfer from a quantum state ϕ to a quantum state ψ can be decomposed into two subsequent transfers

$$\psi \leftarrow \phi' \leftarrow \phi$$

then the resulting amplitude of the transfer is the product of amplitudes of subtransfers: $\langle\psi|\phi\rangle = \langle\psi|\phi'\rangle\langle\phi'|\phi\rangle$

P3 If a transfer from a state ϕ to a state ψ has two independent alternatives



then the resulting amplitude is the sum of amplitudes of two subtransfers.

Hilbert space H_n is n-dimensional complex vector space with

scalar product

$$\langle\psi|\phi\rangle = \sum_{i=1}^{n} \phi_i\psi_i^* \text{of vectors} |\phi\rangle = \begin{vmatrix}\phi_1\\ \phi_2\\ \vdots\\ \phi_n\end{vmatrix}, |\psi\rangle = \begin{vmatrix}\psi_1\\ \psi_2\\ \vdots\\ \psi_n\end{vmatrix},$$

This allows to define the norm of vectors as

$$\|\phi\| = \sqrt{|\langle\phi|\phi\rangle|}.$$

Two vectors $|\phi\rangle$ and $|\psi\rangle$ are called **orthogonal** if $\langle\phi|\psi\rangle = 0$.

A **basis** B of H_n is any set of n vectors $|b_1\rangle, |b_2\rangle, \ldots, |b_n\rangle$ of the norm 1 which are mutually orthogonal.

Given a basis B, any vector $|\psi\rangle$ from H_n can be uniquely expressed in the form

$$|\psi\rangle = \sum_{i=1}^{n} \alpha_i|b_i\rangle.$$

Dirac introduced a very handy notation, so called bra-ket notation, to deal with amplitudes, quantum states and linear functionals $f : H \rightarrow C$.

If $\psi, \phi \in H$, then

$\langle\psi|\phi\rangle$ – scalar product of ψ and ϕ (an amplitude of going from ϕ to ψ).

$|\phi\rangle$ – ket-vector (a column vector) - an equivalent to ϕ

$\langle\psi|$ – bra-vector (a row vector) a linear functional on H

such that $\langle\psi|(|\phi\rangle) = \langle\psi|\phi\rangle$

EVOLUTION in QUANTUM SYSTEM | COMPUTATION in HILBERT SPACE

is described by

Schrödinger linear equation

$$ih\frac{\partial|\Phi(t)\rangle}{\partial t} = H(t)|\Phi(t)\rangle$$

where h is Planck constant, H(t) is a Hamiltonian (total energy) of the system that can be represented by a Hermitian matrix and $\Phi(t)$ is the state of the system in time t.

If the Hamiltonian is time independent then the above Shrödinger equation has solution

$$|\Phi(t)\rangle = U(t)|\Phi(0)\rangle$$

where

$$U(t) = e^{\frac{iHt}{h}}$$

is the evolution operator that can be represented by a unitary matrix. A step of such an evolution is therefore a multiplication of a **unitary matrix** A with a vector $|\psi\rangle$, i.e. A $|\psi\rangle$

A matrix A is **unitary** if

$$A \cdot A^* = A^* \cdot A = I$$

Very important one-qubit unary operators are the following Pauli operators, expressed in the standard basis as follows;

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \sigma_y = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}, \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

PAULI MATRICES

Very important one-qubit unary operators are the following Pauli operators, expressed in the standard basis as follows;

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \sigma_y = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}, \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

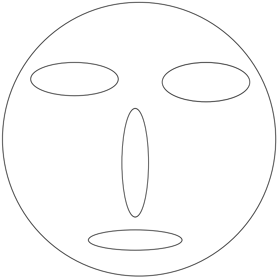
Observe that Pauli matrices transform a qubit state $|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$ as follows

$$\begin{aligned} \sigma_x(\alpha|0\rangle + \beta|1\rangle) &= \beta|0\rangle + \alpha|1\rangle \\ \sigma_z(\alpha|0\rangle + \beta|1\rangle) &= \alpha|0\rangle - \beta|1\rangle \\ \sigma_y(\alpha|0\rangle + \beta|1\rangle) &= \beta|0\rangle - \alpha|1\rangle \end{aligned}$$

Operators σ_x, σ_z and σ_y represent therefore a bit error, a sign error and a bit-sign error.

QUANTUM (PROJECTION) MEASUREMENTS

A quantum state is always observed (measured) with respect to an **observable** O – a decomposition of a given Hilbert space into orthogonal subspaces (where each vector can be uniquely represented as a sum of vectors of these subspaces).



There are two outcomes of a projection measurement of a state $|\phi\rangle$ with respect to O:

1. Classical information into which subspace projection of $|\phi\rangle$ was made.
2. Resulting quantum projection (as a new state) $|\phi'\rangle$ in one of the above subspaces.

The subspace into which projection is made is chosen **randomly** and the corresponding probability is uniquely determined by the amplitudes at the representation of $|\phi\rangle$ as a sum of states of the subspaces.

QUANTUM STATES and PROJECTION MEASUREMENT

In case an orthonormal basis $\{\beta_i\}_{i=1}^n$ is chosen in H_n, any state $|\phi\rangle \in H_n$ can be expressed in the form

$$|\phi\rangle = \sum_{i=1}^{n} a_i|\beta_i\rangle, \sum_{i=1}^{n} |a_i|^2 = 1$$

where

$a_i = \langle\beta_i|\phi\rangle$ are called probability amplitudes

and

their squares provide probabilities

that if the state $|\phi\rangle$ is measured with respect to the basis $\{\beta_i\}_{i=1}^n$, then the state $|\phi\rangle$ collapses into the state $|\beta_i\rangle$ with probability $|a_i|^2$.

The classical "outcome" of a measurement of the state $|\phi\rangle$ with respect to the basis $\{\beta_i\}_{i=1}^n$ is the index i of that state $|\beta_i\rangle$ into which the state collapses.

QUBITS

A qubit is a quantum state in H_2

$$|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$$

where $\alpha, \beta \in C$ are such that $|\alpha|^2 + |\beta|^2 = 1$ and

$\{|0\rangle, |1\rangle\}$ is a (**standard**) basis of H_2

A qubit is a quantum state in H_2

$$|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$$

where $\alpha, \beta \in C$ are such that $|\alpha|^2 + |\beta|^2 = 1$ and

$$\{|0\rangle, |1\rangle\} \text{ is a } (\textbf{standard}) \text{ basis of } H_2$$

EXAMPLE: Representation of qubits by

(a) electron in a Hydrogen atom

(b) a spin-1/2 particle

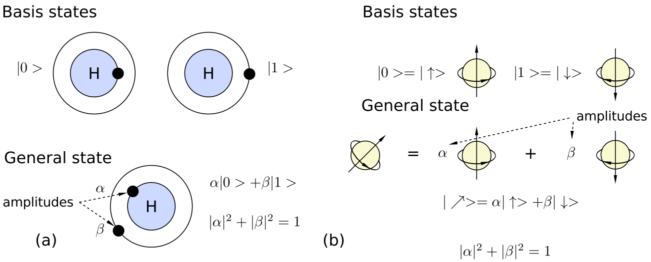


Figure 5: Qubit representations by energy levels of an electron in a hydrogen atom and by a spin-1/2 particle. The condition $|\alpha|^2 + |\beta|^2 = 1$ is a legal one if $|\alpha|^2$ and $|\beta|^2$ are to be the probabilities of being in one of two basis states (of electrons or photons).

STANDARD BASIS

$|0\rangle, |1\rangle$

$$\begin{pmatrix}1\\0\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix}$$

DUAL BASIS

$|0'\rangle, |1'\rangle$

$$\begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{pmatrix}\begin{pmatrix}\frac{1}{\sqrt{2}}\\ -\frac{1}{\sqrt{2}}\end{pmatrix}$$

Hadamard matrix

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$$

$$H|0\rangle = |0'\rangle \qquad H|0'\rangle = |0\rangle$$
$$H|1\rangle = |1'\rangle \qquad H|1'\rangle = |1\rangle$$

transforms one of the basis into another one.

General form of a unitary matrix of degree 2

$$U = e^{i\gamma}\begin{pmatrix}e^{i\alpha} & 0\\ 0 & e^{-i\alpha}\end{pmatrix}\begin{pmatrix}\cos\theta & i\sin\theta\\ i\sin\theta & \cos\theta\end{pmatrix}\begin{pmatrix}e^{i\beta} & 0\\ 0 & e^{-i\beta}\end{pmatrix}$$

of a qubit state

A qubit state can "contain" unboundly large amount of classical information. However, **an unknown quantum state cannot be identified.**

By a **measurement** of the qubit state

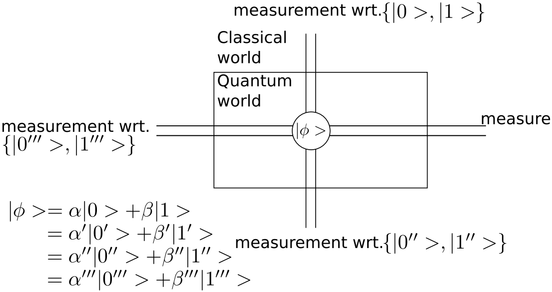
$$\alpha|0\rangle + \beta|1\rangle$$

with respect to the basis

$$\{|0\rangle, |1\rangle\}$$

we can obtain only classical information and only in the following random way:

0 with probability $|\alpha|^2$ 1 with probability $|\beta|^2$



A probability distribution $\{(p_i, |\phi_i\rangle)\}_{i=1}^k$ on pure states is called a mixed state to which it is assigned a density operator

$$\rho = \sum_{i=1}^{n} p_i |\phi\rangle\langle\phi_i|.$$

One interpretation of a mixed state $\{(p_i, |\phi_i\rangle)\}_{i=1}^k$ is that a source X produces the state $|\phi_i\rangle$ with probability p_i.

A probability distribution $\{(p_i, |\phi_i\rangle)\}_{i=1}^k$ on pure states is called a mixed state to which it is assigned a density operator

$$\rho = \sum_{i=1}^{n} p_i |\phi\rangle\langle\phi_i|.$$

One interpretation of a mixed state $\{(p_i, |\phi_i\rangle)\}_{i=1}^k$ is that a source X produces the state $|\phi_i\rangle$ with probability p_i.

Any matrix representing a density operator is called density matrix.

Density matrices are exactly Hermitian, positive matrices with trace 1.

To two different mixed states can correspond the same density matrix.

Two mixes states with the same density matrix are physically undistinguishable.

MAXIMALLY MIXED STATES

To the maximally mixed state

$$\left(\frac{1}{2}, |0\rangle\right), \left(\frac{1}{2}, |1\rangle\right)$$

which represents a random bit corresponds the density matrix

$$\frac{1}{2}\begin{pmatrix}1\\0\end{pmatrix}(1,0) + \frac{1}{2}\begin{pmatrix}0\\1\end{pmatrix}(0,1) = \frac{1}{2}\begin{pmatrix}1 & 0\\0 & 1\end{pmatrix} = \frac{1}{2}I_2$$

MAXIMALLY MIXED STATES

To the maximally mixed state

$$\left(\frac{1}{2}, |0\rangle\right), \left(\frac{1}{2}, |1\rangle\right)$$

which represents a random bit corresponds the density matrix

$$\frac{1}{2}\begin{pmatrix}1\\0\end{pmatrix}(1,0) + \frac{1}{2}\begin{pmatrix}0\\1\end{pmatrix}(0,1) = \frac{1}{2}\begin{pmatrix}1 & 0\\0 & 1\end{pmatrix} = \frac{1}{2}I_2$$

Surprisingly, many other mixed states have density matrix that is the same as that of the maximally mixed state.

QUANTUM ONE-TIME PAD CRYPTOSYSTEM

CLASSICAL ONE-TIME PAD cryptosystem

plaintext	an n-bit string c
shared key	an n-bit string c
cryptotext	an n-bit string c
encoding	$c = p \oplus k$
decoding	$p = c \oplus k$

QUANTUM ONE-TIME PAD CRYPTOSYSTEM

CLASSICAL ONE-TIME PAD cryptosystem

plaintext an n-bit string c
shared key an n-bit string c
cryptotext an n-bit string c
encoding $c = p \oplus k$
decoding $p = c \oplus k$

QUANTUM ONE-TIME PAD cryptosystem

plaintext: an n-qubit string $|p\rangle = |p_1\rangle \ldots |p_n\rangle$
shared key: two n-bit strings k,k'
cryptotext: an n-qubit string $|c\rangle = |c_1\rangle \ldots |c_n\rangle$
encoding: $|c_i\rangle = \sigma_x^{k_i} \sigma_z^{k_i'} |p_i\rangle$
decoding: $|p_i\rangle = \sigma_x^{k_i} \sigma_z^{k_i'} |c_i\rangle$

where $|p_i\rangle = \begin{pmatrix} a_i \\ b_i \end{pmatrix}$ and $|c_i\rangle = \begin{pmatrix} d_i \\ e_i \end{pmatrix}$ are qubits and $\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ with $\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ are Pauli matrices.

In the case of encryption of a qubit

$$|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$$

by QUANTUM ONE-TIME PAD cryptosystem, what is being transmitted is the mixed state

$$\left(\frac{1}{4}, |\phi\rangle\right), \left(\frac{1}{4}, \sigma_x|\phi\rangle\right), \left(\frac{1}{4}, \sigma_z|\phi\rangle\right), \left(\frac{1}{4}, \sigma_x\sigma_z|\phi\rangle\right)$$

whose density matrix is

$$\frac{1}{2}I_2$$

In the case of encryption of a qubit

$$|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$$

by QUANTUM ONE-TIME PAD cryptosystem, what is being transmitted is the mixed state

$$\left(\frac{1}{4}, |\phi\rangle\right), \left(\frac{1}{4}, \sigma_x|\phi\rangle\right), \left(\frac{1}{4}, \sigma_z|\phi\rangle\right), \left(\frac{1}{4}, \sigma_x\sigma_z|\phi\rangle\right)$$

whose density matrix is

$$\frac{1}{2}I_2$$

This density matrix is identical to the density matrix corresponding to that of a random bit, that is to the mixed state

$$\left(\frac{1}{2}, |0\rangle\right), \left(\frac{1}{2}, |1\rangle\right)$$

Shannon classical encryption theorem says that n bits are necessary and sufficient to encrypt securely n bits.

Quantum version of Shannon encryption theorem says that 2n classical bits are necessary and sufficient to encrypt securely n qubits.

COMPOSED QUANTUM SYSTEMS (1)

Tensor product of vectors

$$(x_1,\dots,x_n)\otimes(y_1,\dots,y_m)=(x_1y_1,\dots,x_1y_m,x_2y_1,\dots,x_2y_m,\dots,x_2y_m,\dots,x_ny_1,\dots,x_ny_m)$$

Tensor product of matrices $A\otimes B=\begin{pmatrix} a_{11}B & \dots & a_{1n}B \\ \vdots & & \vdots \\ a_{n1}B & \dots & a_{nn}B \end{pmatrix}$

where $A=\begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nn} \end{pmatrix}$

COMPOSED QUANTUM SYSTEMS (1)

Tensor product of vectors

$$(x_1,\dots,x_n)\otimes(y_1,\dots,y_m)=(x_1y_1,\dots,x_1y_m,x_2y_1,\dots,x_2y_m,\dots,x_2y_m,\dots,x_ny_1,\dots,x_ny_m)$$

Tensor product of matrices $A\otimes B=\begin{pmatrix} a_{11}B & \dots & a_{1n}B \\ \vdots & & \vdots \\ a_{n1}B & \dots & a_{nn}B \end{pmatrix}$

where $A=\begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nn} \end{pmatrix}$

Example $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\otimes\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}=\begin{pmatrix} a_{11} & a_{12} & 0 & 0 \\ a_{21} & a_{22} & 0 & 0 \\ 0 & 0 & a_{11} & a_{12} \\ 0 & 0 & a_{21} & a_{22} \end{pmatrix}$

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}\otimes\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}=\begin{pmatrix} a_{11} & 0 & a_{12} & 0 \\ 0 & a_{11} & 0 & a_{12} \\ a_{21} & 0 & a_{22} & 0 \\ 0 & a_{21} & 0 & a_{22} \end{pmatrix}$$

COMPOSED QUANTUM SYSTEMS (2)

Tensor product of Hilbert spaces $H_1 \otimes H_2$ is the complex vector space spanned by tensor products of vectors from H_1 and H_2 . That corresponds to the quantum system composed of the quantum systems corresponding to Hilbert spaces H_1 and H_2.

An important difference between classical and quantum systems

A state of a compound classical (quantum) system can be (cannot be) always composed from the states of the subsystem.

QUANTUM REGISTERS

A general state of a 2-qubit register is:

$$|\phi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$$

where

$$|\alpha_{00}|^2 + |\alpha_{01}|^2 + |\alpha_{10}|^2 + |\alpha_{11}|^2 = 1$$

and $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ are vectors of the "standard" basis of H_4, i.e.

$$|00\rangle = \begin{pmatrix}1\\0\\0\\0\end{pmatrix} |01\rangle = \begin{pmatrix}0\\1\\0\\0\end{pmatrix} |10\rangle = \begin{pmatrix}0\\0\\1\\0\end{pmatrix} |11\rangle = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}$$

An important unitary matrix of degree 4, to transform states of 2-qubit registers:

$$CNOT = XOR = \begin{pmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 1 & 0\\0 & 0 & 0 & 1\end{pmatrix}$$

It holds:

$$\text{CNOT} : |x, y\rangle \Rightarrow |x, x \oplus y\rangle$$

of the states of 2-qubit registers

$$|\phi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$$

1. Measurement with respect to the basis $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$
   RESULTS:

$|00\rangle$ and 00 with probability $|\alpha_{00}|^2$
$|01\rangle$ and 01 with probability $|\alpha_{01}|^2$
$|10\rangle$ and 10 with probability $|\alpha_{10}|^2$
$|11\rangle$ and 11 with probability $|\alpha_{11}|^2$

QUANTUM MEASUREMENT

of the states of 2-qubit registers

$$|\phi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$$

1. Measurement with respect to the basis $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$
   RESULTS:

   $|00\rangle$ and 00 with probability $|\alpha_{00}|^2$
   $|01\rangle$ and 01 with probability $|\alpha_{01}|^2$
   $|10\rangle$ and 10 with probability $|\alpha_{10}|^2$
   $|11\rangle$ and 11 with probability $|\alpha_{11}|^2$

2. Measurement of particular qubits:
   By measuring the first qubit we get

   0 with probability $|\alpha_{00}|^2 + |\alpha_{01}|^2$

   and $|\phi\rangle$ is reduced to the vector $\dfrac{\alpha_{00}|00\rangle + \alpha_{01}|01\rangle}{\sqrt{|\alpha_{10}|^2 + |\alpha_{11}|^2}}$

   1 with probability $|\alpha_{10}|^2 + |\alpha_{11}|^2$

   and $|\phi\rangle$ is reduced to the vector $\dfrac{\alpha_{10}|10\rangle + \alpha_{11}|11\rangle}{\sqrt{|\alpha_{10}|^2 + |\alpha_{11}|^2}}$

NO-CLONING THEOREM

INFORMAL VERSION: Unknown quantum state cannot be cloned.

NO-CLONING THEOREM

INFORMAL VERSION: Unknown quantum state cannot be cloned.

FORMAL VERSION: There is no unitary transformation U such that for any qubit state $|\psi\rangle$

$$U(|\psi\rangle|0\rangle) = |\psi\rangle|\psi\rangle$$

NO-CLONING THEOREM

INFORMAL VERSION: Unknown quantum state cannot be cloned.

FORMAL VERSION: There is no unitary transformation U such that for any qubit state $|\psi\rangle$

$$U(|\psi\rangle|0\rangle) = |\psi\rangle|\psi\rangle$$

PROOF: Assume U exists and for two different states $|\alpha\rangle$ and $|\beta\rangle$

$$U(|\alpha\rangle|0\rangle) = |\alpha\rangle|\alpha\rangle \qquad U(|\beta\rangle|0\rangle) = |\beta\rangle|\beta\rangle$$

Let

$$|\gamma\rangle = \frac{1}{\sqrt{2}}(|\alpha\rangle + |\beta\rangle)$$

Then

$$U(|\gamma\rangle|0\rangle) = \frac{1}{\sqrt{2}}(|\alpha\rangle|\alpha\rangle + |\beta\rangle|\beta\rangle) \neq |\gamma\rangle|\gamma\rangle = \frac{1}{\sqrt{2}}(|\alpha\rangle|\alpha\rangle + |\beta\rangle|\beta\rangle + |\alpha\rangle|\beta\rangle + |\beta\rangle|\alpha\rangle)$$

However, CNOT can make copies of basis states $|0\rangle, |1\rangle$:

$$CNOT(|x\rangle|0\rangle) = |x\rangle|x\rangle$$

BELL STATES

States

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle), \qquad |\Phi^-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$$
$$|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle), \qquad |\Psi^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$$

form an orthogonal (Bell) basis in H_4 and play an important role in quantum computing.

Theoretically, there is an observable for this basis. However, no one has been able to construct a measuring device for Bell measurement using linear elements only.

QUANTUM n-qubit REGISTER

A general state of an n-qubit register has the form:

$$|\phi\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle = \sum_{i \in \{0,1\}^n} \alpha_i |i\rangle, \text{ where } \sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1$$

and $|\phi\rangle$ is a vector in H_{2^n}.

Operators on n-qubits registers are unitary matrices of degree 2^n.

Is it difficult to create a state of an n-qubit register?

In general yes, in some important special cases not. For example, if n-qubit Hadamard transformation

$$H_n = \otimes_{i=1}^n H.$$

is used then

$$H_n|0^{(n)}\rangle = \otimes_{i=1}^n H|0\rangle = \otimes_{i=1}^n |0'\rangle = |0'^{(n)}\rangle = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle$$

and, in general, for $x \in \{0,1\}^n$

$$H_n|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle.$$ [1]

[1] The dot product is defined as follows: $x \cdot y = \otimes_{i=1}^n x_i y_i$.

QUANTUM PARALLELISM

If

$$f : \{0, 1, \ldots, 2^n - 1\} \Rightarrow \{0, 1, \ldots, 2^n - 1\}$$

then the mapping

$$f' : (x, 0) \Rightarrow (x, f(x))$$

is one-to-one and therefore there is a unitary transformation U_f such that.

$$U_f(|x\rangle|0\rangle) \Rightarrow |x\rangle|f(x)\rangle$$

Let us have the state

$$|\Psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle|0\rangle$$

With a single application of the mapping U_f we then get

$$U_f|\Psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle|f(i)\rangle$$

OBSERVE THAT IN A SINGLE COMPUTATIONAL STEP 2^n VALUES OF f ARE COMPUTED!

IN WHAT LIES POWER OF QUANTUM COMPUTING?

In quantum superposition or in quantum parallelism?
NOT,
in QUANTUM ENTANGLEMENT!

Let

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

be a state of two very distant particles, **for example** on two planets
Measurement of one of the particles, with respect to the standard basis, makes the above state to collapse to one of the states

$$|00\rangle \text{ or } |11\rangle.$$

This means that subsequent measurement of other particle (on another planet) provides the same result as the measurement of the first particle. This indicate that in quantum world non-local influences, correlations, exist.

POWER of ENTANGLEMENT

Quantum state $|\Psi\rangle$ of a composed bipartite quantum system $A \otimes B$ is called entangled if it cannot be decomposed into tensor product of the states from A and B.

Quantum entanglement is an important quantum resource that allows

- To create phenomena that are impossible in the classical world (for example teleportation)
- To create quantum algorithms that are asymptotically more efficient than any classical algorithm known for the same problem.
- To create communication protocols that are asymptotically more efficient than classical communication protocols for the same task
- To create, for two parties, shared secret binary keys
- To increase capacity of quantum channels

CLASSICAL versus QUANTUM CRYPTOGRAPHY

- Security of classical cryptography is based on unproven assumptions of computational complexity (and it can be jeopardize by progress in algorithms and/or technology).

  Security of quantum cryptography is based on laws of quantum physics that allow to build systems where undetectable eavesdropping is impossible.

- Security of classical cryptography is based on unproven assumptions of computational complexity (and it can be jeopardize by progress in algorithms and/or technology).

  Security of quantum cryptography is based on laws of quantum physics that allow to build systems where undetectable eavesdropping is impossible.

- Since classical cryptography is vulnerable to technological improvements it has to be designed in such a way that a secret is secure with respect to **future technology**, during the whole period in which the secrecy is required.

  Quantum key generation, on the other hand, needs to be designed only to be secure against technology available at the moment of key generation.

QUANTUM KEY GENERATION

Quantum protocols for using quantum systems to achieve unconditionally secure generation of secret (classical) keys by two parties are one of the main theoretical achievements of quantum information processing and communication research.

Moreover, experimental systems for implementing such protocols are one of the main achievements of experimental quantum information processing research.

It is believed and hoped that it will be

quantum key generation (QKG)

another term is

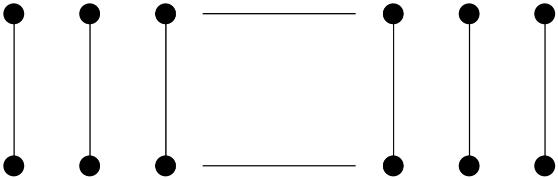
quantum key distribution (QKD)

where one can expect the first

transfer from the experimental to the development stage.

Let Alice and Bob share n pairs of particles in the entangled EPR-state.

$$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$



n pairs of particles in EPR state

If both of them measure their particles in the standard basis, then they get, as the classical outcome of their measurements the same random, shared and secret binary key of length n.

POLARIZATION of PHOTONS

Polarized photons are currently mainly used for experimental quantum key generation.

Photon, or light quantum, is a particle composing light and other forms of electromagnetic radiation.

Photons are electromagnetic waves and their electric and magnetic fields are perpendicular to the direction of propagation and also to each other.

An important property of photons is polarization – it refers to the bias of the electric field in the electromagnetic field of the photon.

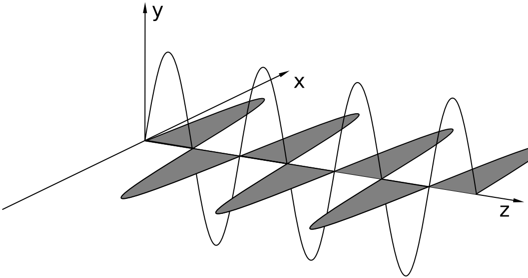


Figure 6: Electric and magnetic fields of a linearly polarized photon

POLARIZATION of PHOTONS

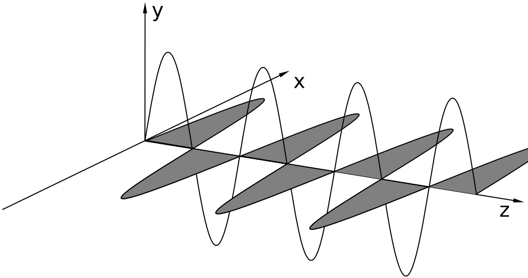


Figure 6: Electric and magnetic fields of a linearly polarized photon

If the electric field vector is always parallel to a fixed line we have **linear polarization (see Figure).**

POLARIZATION of PHOTONS

There is no way to determine exactly polarization of a single photon.

However, for any angle θ there are θ-**polarizers** – "filters" – that produce θ-polarized photons from an incoming stream of photons and they let θ_1-polarized photons to get through with probability $cos^2(\theta - \theta_1)$.

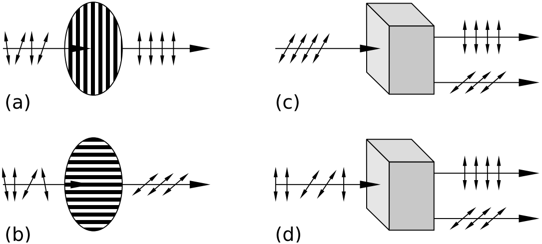


Figure 6: Photon polarizers and measuring devices-80%

Photons whose electronic fields oscillate in a plane at either 0° or 90° to some reference line are called usually **rectilinearly polarized** and those whose electric field oscillates in a plane at 45° or 135° as **diagonally polarized**. Polarizers that produce only vertically or horizontally polarized photons are depicted in Figure 6 a, b.

Generation of orthogonally polarized photons.

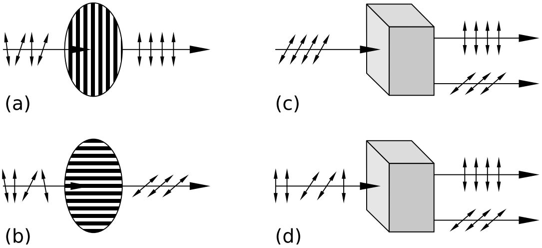


Figure 6: Photon polarizers and measuring devices-80%

For any two orthogonal polarizations there are generators that produce photons of two given orthogonal polarizations. For example, a calcite crystal, properly oriented, can do the job.

Fig. c – a calcite crystal that makes θ-polarized photons to be horizontally (vertically) polarized with probability $cos^2\theta(sin^2\theta)$.

Fig. d – a calcite crystal can be used to separate horizontally and vertically polarized photons.

Very basic setting Alice tries to send a quantum system to Bob and an eavesdropper tries to learn, or to change, as much as possible, without being detected.

Eavesdroppers have this time especially hard time, because quantum states cannot be copied and cannot be measured without causing, in general, a disturbance.

Key problem: Alice prepares a quantum system in a specific way, unknown to the eavesdropper, Eve, and sends it to Bob.

The question is how much information can Eve extract of that quantum system and how much it costs in terms of the disturbance of the system.

Three special cases

1. Eve has no information about the state $|\psi\rangle$ Alice sends.
2. Eve knows that $|\psi\rangle$ is one of the states of an orthonormal basis $\{|\phi_i\rangle\}_{i=1}^{n}$.
3. Eve knows that $|\psi\rangle$ is one of the states $|\phi_1\rangle, \ldots, |\phi_n\rangle$ that **are not** mutually orthonormal and that p_i is the probability that $|\psi\rangle = |\phi_i\rangle$.

TRANSMISSION ERRORS

If Alice sends randomly chosen bit

$$0 \text{ encoded randomly as } |0\rangle \text{ or } |0'\rangle$$

or

$$1 \text{ encoded as randomly as } |1\rangle \text{ or } |1'\rangle$$

and Bob measures the encoded bit by choosing randomly the standard or the dual basis, then the probability of error is $\frac{1}{4} = \frac{2}{8}$

If Eve measures the encoded bit, sent by Alice, according to the randomly chosen basis, standard or dual, then she can learn the bit sent with the probability 75% .

If she then sends the state obtained after the measurement to Bob and he measures it with respect to the standard or dual basis, randomly chosen, then the probability of error for his measurement is $\frac{3}{8}$ – a 50% increase with respect to the case there was no eavesdropping.

Indeed the error is

$$\frac{1}{2} \cdot \frac{1}{4} + \frac{1}{2}\Big(\frac{1}{2} \cdot \frac{1}{4} + \frac{1}{2} \cdot \frac{3}{4}\Big) = \frac{3}{8}$$

BB84 QUANTUM KEY GENERATION PROTOCOL

Quantum key generation protocol BB84 (due to Bennett and Brassard), for generation of a key of length n, has several phases:

Preparation phase

Quantum key generation protocol BB84 (due to Bennett and Brassard), for generation of a key of length n, has several phases:

Preparation phase

Alice is assumed to have four transmitters of photons in one of the following four polarizations 0, 45, 90 and 135 degrees

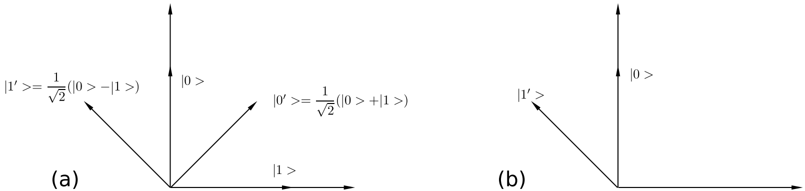


Figure 8: Polarizations of photons for BB84 and B92 protocols

Expressed in a more general form, Alice uses for encoding states from the set $\{|0\rangle, |1\rangle, |0'\rangle, |1'\rangle\}$.
Bob has a detector that can be set up to distinguish between rectilinear polarizations (0 and 90 degrees) or can be quickly reset to distinguish between diagonal polarizations (45 and 135 degrees).

BB84 QUANTUM KEY GENERATION PROTOCOL

(In accordance with the laws of quantum physics, there is no detector that could distinguish between unorthogonal polarizations.)

(In a more formal setting, Bob can measure the incomming photons either in the standard basis $B = \{|0\rangle, |1\rangle\}$ or in the dual basis $D = \{|0'\rangle, |1'\rangle\}$.

To send a bit 0 (1) of her first random sequence through a quantum channel Alice chooses, on the basis of her second random sequence, one of the encodings $|0\rangle$ or $|0'\rangle$ ($|1\rangle$ or $|1'\rangle$), i.e., in the standard or dual basis,

Bob chooses, each time on the base of his private random sequence, one of the bases B or D to measure the photon he is to receive and he records the results of his measurements and keeps them secret.

Alice's encodings	Bob's observables	Alice's state relative to Bob	The result and its probability	Correctness
$0 \rightarrow \lvert 0\rangle$	$0 \rightarrow B$	$\lvert 0\rangle$	0 (prob. 1)	correct
	$1 \rightarrow D$	$\frac{1}{\sqrt{2}}(\lvert 0'\rangle + \lvert 1'\rangle)$	0/1 (prob. $\frac{1}{2}$)	random
$0 \rightarrow \lvert 0'\rangle$	$0 \rightarrow B$	$\frac{1}{\sqrt{2}}(\lvert 0\rangle + \lvert 1\rangle)$	0/1 (prob. $\frac{1}{2}$)	random
	$1 \rightarrow D$	$\lvert 0'\rangle$	0 (prob. 1)	correct
$1 \rightarrow \lvert 1\rangle$	$0 \rightarrow B$	$\lvert 1\rangle$	1 (prob. 1)	correct
	$1 \rightarrow D$	$\frac{1}{\sqrt{2}}(\lvert 0'\rangle - \lvert 1'\rangle)$	0/1 (prob. $\frac{1}{2}$)	random
$1 \rightarrow \lvert 1'\rangle$	$0 \rightarrow B$	$\frac{1}{\sqrt{2}}(\lvert 0\rangle + \lvert 1\rangle)$	0/1 (prob. $\frac{1}{2}$)	random
	$1 \rightarrow D$	$\lvert 1'\rangle$	1 (prob. 1)	correct

Figure 9: Quantum cryptography with BB84 protocol

Figure 9 shows the possible results of the measurements and their probabilities.

An example of an encoding – decoding process is in the Figure 10.

Raw key extraction

Bob makes public the sequence of bases he used to measure the photons he received – but not the results of the measurements – and Alice tells Bob, through a classical channel, in which cases he has chosen the same basis for measurement as she did for encoding. The corresponding bits then form the basic raw key.

1	0	0	0	1	1	0	0	0	1	1	Alice's random sequence
$\vert 1\rangle$	$\vert 0'\rangle$	$\vert 0\rangle$	$\vert 0'\rangle$	$\vert 1\rangle$	$\vert 1'\rangle$	$\vert 0'\rangle$	$\vert 0\rangle$	$\vert 0\rangle$	$\vert 1\rangle$	$\vert 1'\rangle$	Alice's polarizations
0	1	1	1	0	0	1	0	0	1	0	Bob's random sequence
B	D	D	D	B	B	D	B	B	D	B	Bob's observable
1	0	R	0	1	R	0	0	0	R	R	outcomes

Figure 10: Quantum transmissions in the BB84 protocol – R stands for the case that the result of the measurement is random.

Test for eavesdropping

Alice and Bob agree on a sequence of indices of the raw key and make the corresponding bits of their raw keys public.

Case 1. Noiseless channel. If the subsequences chosen by Alice and Bob are not completely identical eavesdropping is detected. Otherwise, the remaining bits are taken as creating the final key.

Case 2. Noisy channel. If the subsequences chosen by Alice and Bob contains more errors than the admitable error of the channel (that has to be determined from channel characteristics), then eavesdropping is assumed. Otherwise, the remaining bits are taken as the next result of the raw key generation process.

Test for eavesdropping

Alice and Bob agree on a sequence of indices of the raw key and make the corresponding bits of their raw keys public.

Case 1. Noiseless channel. If the subsequences chosen by Alice and Bob are not completely identical eavesdropping is detected. Otherwise, the remaining bits are taken as creating the final key.

Case 2. Noisy channel. If the subsequences chosen by Alice and Bob contains more errors than the admitable error of the channel (that has to be determined from channel characteristics), then eavesdropping is assumed. Otherwise, the remaining bits are taken as the next result of the raw key generation process.

Error correction phase

In the case of a noisy channel for transmission it may happen that Alice and Bob have different raw keys after the key generation phase.

A way out is to use a special error correction techniques and at the end of this stage both Alice and Bob share identical keys.

Privacy amplification phase

One problem remains. Eve can still have quite a bit of information about the key both Alice and Bob share. Privacy amplification is a tool to deal with such a case.

Privacy amplification is a method how to select a short and very secret binary string s from a longer but less secret string s'. The main idea is simple. If $|s| = n$, then one picks up n random subsets $S_1, \ldots, S_n$ of bits of s' and let s_i, the i-th bit of S, be the parity of S_i. One way to do it is to take a random binary matrix of size $|s| \times |s'|$ and to perform multiplication Ms'^T, where s'^T is the binary column vector corresponding to s'.

The point is that even in the case where an eavesdropper knows quite a few bits of s', she will have almost no information about s.

More exactly, if Eve knows parity bits of k subsets of s', then if a random subset of bits of s' is chosen, then the probability that Eve has any information about its parity bit is less than $\dfrac{2^{-(n-k-1)}}{\ln 2}$.

Successes

1. Transmissions using optical fibers to the distance of 120 km.
2. Open air transmissions to the distance 144 km at day time (from one pick of Canary Islands to another).
3. Next goal: earth to satellite transmissions.

Successes

1. Transmissions using optical fibers to the distance of 120 km.
2. Open air transmissions to the distance 144 km at day time (from one pick of Canary Islands to another).
3. Next goal: earth to satellite transmissions.

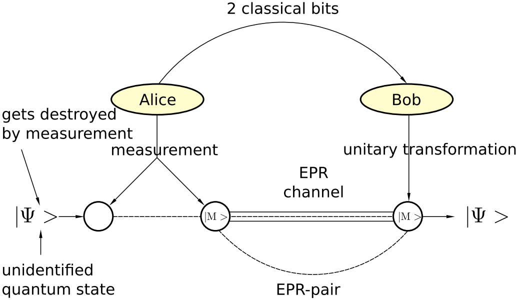
All current systems use optical means for quantum state transmissions

Problems and tasks

1. No single photon sources are available. Weak laser pulses currently used contains in average 0.1 - 0.2 photons.
2. Loss of signals in the fiber. (Current error rates: 0,5 - 4%)
3. To move from the experimental to the developmental stage.

QUANTUM TELEPORTATION

Quantum teleportation allows to transmit unknown quantum information to a very distant place in spite of impossibility to measure or to broadcast information to be transmitted.



$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \qquad |EPR - pair\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

Total state

$$|\psi\rangle|EPR - pair\rangle = \frac{1}{\sqrt{2}}(\alpha|000\rangle + \alpha|011\rangle + \beta|100\rangle + \beta|111\rangle)$$

Measurement of the first two qubits is done with respect to the "Bell basis":

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \qquad |\Phi^-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$$

$$|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \qquad |\Psi^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$$

QUANTUM TELEPORTATION I

Total state of three particles:

$$|\psi\rangle|EPR - pair\rangle = \frac{1}{\sqrt{2}}(\alpha|000\rangle + \alpha|011\rangle + \beta|100\rangle + \beta|111\rangle)$$

can be expressed as follows:

$$|\psi\rangle|EPR - pair\rangle = |\Phi^+\rangle\frac{1}{\sqrt{2}}(\alpha|0\rangle + \beta|1\rangle) + |\Psi^+\rangle\frac{1}{\sqrt{2}}(\beta|0\rangle + \alpha|1\rangle) + |\Phi^-\rangle\frac{1}{\sqrt{2}}(\alpha|0\rangle - \beta|1\rangle) + |\Psi^-\rangle\frac{1}{\sqrt{2}}(-\beta|0\rangle + \alpha|1\rangle)$$

and therefore Bell measurement of the first two particles projects the state of Bob's particle into a "small modification" $|\psi_1\rangle$ of the state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$,

$$|\Psi_1\rangle = \text{either } |\Psi\rangle \text{ or } \sigma_x|\Psi\rangle \text{ or } \sigma_z|\Psi\rangle \text{ or } \sigma_x\sigma_z|\psi\rangle$$

The unknown state $|\psi\rangle$ can therefore be obtained from $|\psi_1\rangle$ by applying one of the four operations

$$\sigma_x, \sigma_y, \sigma_z, I$$

and the result of the Bell measurement provides two bits specifying which of the above four operations should be applied.

These four bits Alice needs to send to Bob using a classical channel (by email, for example).

QUANTUM TELEPORTATION II

If the first two particles of the state

$$|\psi\rangle|EPR-pair\rangle = |\Phi^+\rangle\frac{1}{\sqrt{2}}(\alpha|0\rangle+\beta|1\rangle) + |\Psi^+\rangle\frac{1}{\sqrt{2}}(\beta|0\rangle+\alpha|1\rangle) + |\Phi^-\rangle\frac{1}{\sqrt{2}}(\alpha|0\rangle - \beta|1\rangle) + |\Psi^-\rangle\frac{1}{\sqrt{2}}(-\beta|0\rangle+\alpha|1\rangle)$$

are measured with respect to the Bell basis then Bob's particle gets into the mixed state

$$\left(\frac{1}{4},\alpha|0\rangle+\beta|1\rangle\right)\oplus\left(\frac{1}{4},\alpha|0\rangle-\beta|1\rangle\right)\oplus\left(\frac{1}{4},\beta|0\rangle+\alpha|1\rangle\right)\oplus\left(\frac{1}{4},\beta|0\rangle-\alpha|1\rangle\right)$$

to which corresponds the density matrix

$$\frac{1}{4}\binom{\alpha^*}{\beta^*}(\alpha,\beta)+\frac{1}{4}\binom{\alpha^*}{-\beta^*}(\alpha,-\beta)+\frac{1}{4}\binom{\beta^*}{\alpha^*}(\beta,\alpha)+\frac{1}{4}\binom{\beta^*}{-\alpha^*}(\beta,-\alpha)=\frac{1}{2}I$$

The resulting density matrix is identical to the density matrix for the mixed state

$$\left(\frac{1}{2},|0\rangle\right)\oplus\left(\frac{1}{2},|1\rangle\right)$$

Indeed, the density matrix for the last mixed state has the form

$$\frac{1}{2}\binom{1}{0}(1,0)+\frac{1}{2}\binom{0}{1}(0,1)=\frac{1}{2}I$$

QUANTUM TELEPORTATION – COMMENTS

- Alice can be seen as dividing information contained in $|\psi\rangle$ into
  - quantum information – transmitted through EPR channel
  - classical information – transmitted through a classical channel

QUANTUM TELEPORTATION – COMMENTS

- Alice can be seen as dividing information contained in $|\psi\rangle$ into
  - quantum information – transmitted through EPR channel
  - classical information – transmitted through a classical channel
- In a quantum teleportation an unknown quantum state $|\phi\rangle$ can be disassembled into, and later reconstructed from, two classical bit-states and an maximally entangled pure quantum state.

QUANTUM TELEPORTATION – COMMENTS

- Alice can be seen as dividing information contained in $|\psi\rangle$ into
  - quantum information – transmitted through EPR channel
  - classical information – transmitted through a classical channel
- In a quantum teleportation an unknown quantum state $|\phi\rangle$ can be disassembled into, and later reconstructed from, two classical bit-states and an maximally entangled pure quantum state.
- Using quantum teleportation an unknown quantum state can be teleported from one place to another by a sender who does need to know – for teleportation itself – neither the state to be teleported nor the location of the intended receiver.

QUANTUM TELEPORTATION – COMMENTS

- Alice can be seen as dividing information contained in $|\psi\rangle$ into
  - quantum information – transmitted through EPR channel
  - classical information – transmitted through a classical channel
- In a quantum teleportation an unknown quantum state $|\phi\rangle$ can be disassembled into, and later reconstructed from, two classical bit-states and an maximally entangled pure quantum state.
- Using quantum teleportation an unknown quantum state can be teleported from one place to another by a sender who does need to know – for teleportation itself – neither the state to be teleported nor the location of the intended receiver.
- The teleportation procedure can not be used to transmit information faster than light

  but

  it can be argued that quantum information presented in unknown state is transmitted instantaneously (except two random bits to be transmitted at the speed of light at most).

- Alice can be seen as dividing information contained in $|\psi\rangle$ into
  - quantum information – transmitted through EPR channel
  - classical information – transmitted through a classical channel
- In a quantum teleportation an unknown quantum state $|\phi\rangle$ can be disassembled into, and later reconstructed from, two classical bit-states and an maximally entangled pure quantum state.
- Using quantum teleportation an unknown quantum state can be teleported from one place to another by a sender who does need to know – for teleportation itself – neither the state to be teleported nor the location of the intended receiver.
- The teleportation procedure can not be used to transmit information faster than light

  but

  it can be argued that quantum information presented in unknown state is transmitted instantaneously (except two random bits to be transmitted at the speed of light at most).
- EPR channel is irreversibly destroyed during the teleportation process.

- In Cambridge connecting Harvard, Boston Uni, and BBN Technology (10,19 and 29 km).
- Currently 6 nodes, in near future 10 nodes.
- Continuously operating since March 2004
- Three technologies: lasers through optic fibers, entanglement through fiber and free-space QKD (in future two versions of it).
- Implementation of BB84 with authentication, sifting error correction and privacy amplification.
- One 2x2 switch to make sender-receiver connections
- Capability to overcome several limitations of stand-alone QKD systems.

WHY IS QUANTUM INFORMATION PROCESSING SO IMPORTANT

- QIPC is believed to lead to new Quantum Information Processing Technology that could have broad impacts.
- Several areas of science and technology are approaching such points in their development where they badly need expertise with storing, transmission and processing of particles.
- It is increasingly believed that new, quantum information processing based, understanding of (complex) quantum phenomena and systems can be developed.
- Quantum cryptography seems to offer new level of security and be soon feasible.
- QIPC has been shown to be more efficient in interesting/important cases.

The main task at quantum computation is to express solution of a given problem P as a unitary matrix U and then to construct a circuit C_U with elementary quantum gates from a universal sets of quantum gates to realize U.

The main task at quantum computation is to express solution of a given problem P as a unitary matrix U and then to construct a circuit C_U with elementary quantum gates from a universal sets of quantum gates to realize U.

A simple universal set of quantum gates consists of gates.

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \sigma_z^{\frac{1}{4}} = \begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{\pi}{4} i} \end{pmatrix}$$

FUNDAMENTAL RESULTS

The first really satisfactory results, concerning universality of gates, have been due to Barenco et al. (1995)

Theorem 0.1 CNOT gate and all one-qubit gates form a universal set of gates.

The proof is in principle a simple modification of the RQ-decomposition from linear algebra. Theorem 0.1 can be easily improved:

Theorem 0.2 CNOT gate and elementary rotation gates

$$R_\alpha(\theta) = \cos\frac{\theta}{2} I - i \sin\frac{\theta}{2}\sigma_\alpha \qquad \text{for } \alpha \in \{x, y, z\}$$

form a universal set of gates.

Quantum algorithms are methods of using quantum circuits and processors to solve algorithmic problems.

On a more technical level, a design of a quantum algorithm can be seen as a process of an efficient decomposition of a complex unitary transformation into products of elementary unitary operations (or gates), performing simple local changes.

Quantum algorithms are methods of using quantum circuits and processors to solve algorithmic problems.

On a more technical level, a design of a quantum algorithm can be seen as a process of an efficient decomposition of a complex unitary transformation into products of elementary unitary operations (or gates), performing simple local changes.

The four main features of quantum mechanics that are exploited in quantum computation:

- Superposition;
- Interference;
- Entanglement;
- Measurement.

EXAMPLES of QUANTUM ALGORITHMS

Deutsch problem: Given is a black-box function f: $\{0,1\} \rightarrow \{0,1\}$, how many queries are needed to find out whether f is constant or balanced:

Classically: 2

Quantumly: 1

Deutsch-Jozsa Problem: Given is a black-box function $f : \{0,1\}^n \rightarrow \{0,1\}$ and a promise that f is either constant or balanced, how many queries are needed to find out whether f is constant or balanced.

Classically: n

Quantumly 1

Factorization of integers: all classical algorithms are exponential.

Peter Shor developed polynomial time quantum algorithm

Search of an element in an unordered database of n elements:

Classically n queries are needed in the worst case

Lov Grover showed that quantumly $\sqrt{n}$ queries are enough