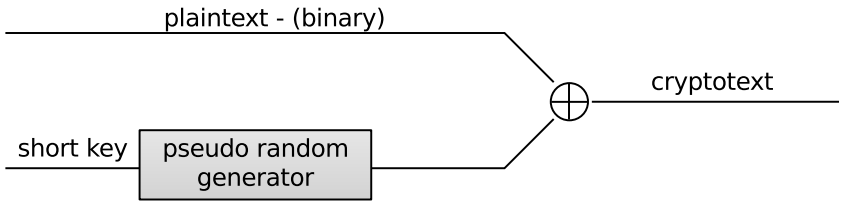
Part XII

From theory to practice in cryptography

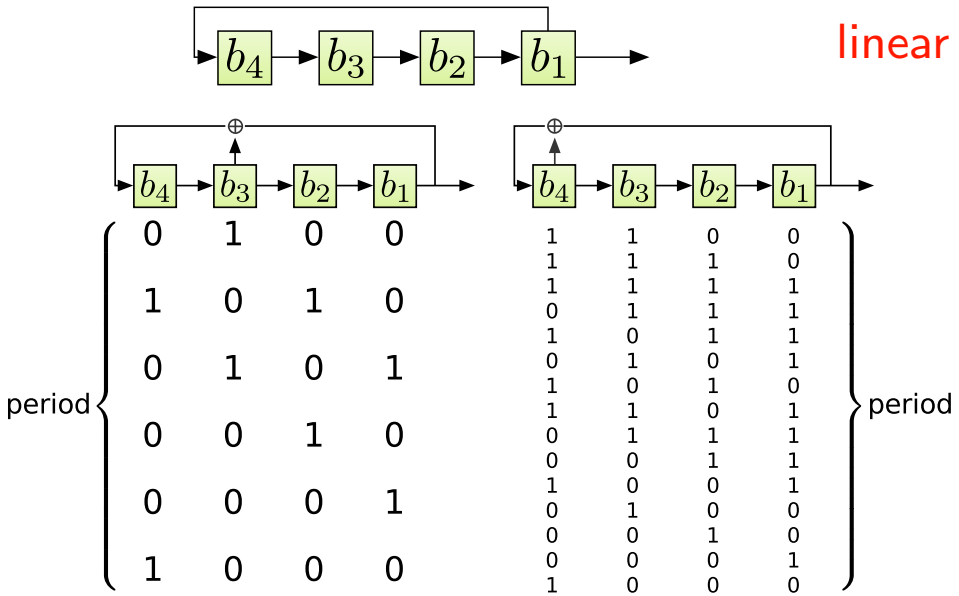
I.SHIFT REGISTERS

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

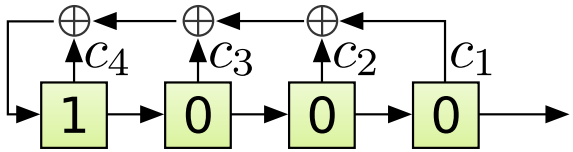
Theorem For every $n > 0$ there is a linear shift register of maximal period $2^n - 1$.

CRYPTOANALYSIS 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.

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 outputed 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 Recurrences

Linear feedback shift registers are an efficient way to realize recurrence relations of the type

$$x_{n+m} = c_0 x_n + c_1 x_{n+1} + \cdots + c_{m-1} x_{n+m-1} \pmod{n}$$

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.

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.

Finding Linear Recurrences – a method

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.

Finding Linear Recurrences

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 cryptoanalyst'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 difusion – a more detailed view

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.

III. Cryptosystem DES – its history

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 a 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!!!!!!

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 $c = \phi_{64}^{-1}(L_{16}, R_{16})$

DES cryptosystem – Data Encryption Standard – 1977

Decryption $\phi_{64}(c) = L_{16}R_{16}$ is computed and then the recurrence

$$R_{i-1} = L_i$$
$$L_{i-1} = R_i \oplus f(L_i, K_i),$$

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 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.
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 sugested that for $ 20 milions 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.

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.

Weaknesses of DES

- 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.

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 choosen, $z_i = e_k(z_{i-1})$ 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:

$$C_j = P_j \oplus L_8(e_k(X_j))$$
$$X_{j+1} = R_{56}(X_j) \| C_j,$$

$L_8(X)$ 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:

$$P_j = C_j \oplus L_8(e_k(X_j))$$
$$X_{j+1} = R_{56}(X_j) \| C_j,$$

Advantages of different encryption modes

- 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.

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 1999 they did that in 24 hours.
- It started to be clear that a new cryptosystem with larger keys is badly needed.

Product- and Feistel-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.

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 cryptosystem

- 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.

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

ARITHMETICS 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 bitwise-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

  $$(x^6 + x^4 + x + 1) + (x^7 + x^2 + x + 1) = x^7 + x^6 + x^4 + x^2$$

- $$\text{`57'} \bullet \text{`83'} = \text{`C1'}$$

  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

  $$\begin{aligned} \text{`57'} \bullet \text{`02'} &= \text{xtime}(57) = \text{`AE'} \\ \text{`57'} \bullet \text{`04'} &= \text{xtime}(\text{AE}) = \text{`47'} \\ \text{`57'} \bullet \text{`08'} &= \text{xtime}(47) = \text{`8E'} \\ \text{`57'} \bullet \text{`10'} &= \text{xtime}(8\text{E}) = \text{`07'} \end{aligned}$$

POLYNOMIALS over GF(2^8)

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 b = 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 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.

KEY EXPANSION

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

$$\mathrm{V} = \begin{cases} F(W_{i-1}), & \text{if i mod k} = 0 \\ G(W_{i-1}), & \text{if i mod k} = 4 \text{ and } D_k = 256 \text{ bits}, \\ W_{i-1} & \text{otherwise} \end{cases}$$

where the function G performs only the byte-substitution of the corresponding bytes. **F**unction F is defined in a quite a complicated way.

STEPS of ENCRYPTION

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

$$\begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix}$$

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
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.

Pronounciation 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.

Digital envelopes

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 decription 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?

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.

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 tamperproof 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.
- 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 revocated 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 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).

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;

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.

Pretty Good Privacy

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.

How cryptographic systems get broken

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.

RSA in practice

- 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.