

## CHAPTER 9: User identification and message authentication, Secret sharing and E-commerce

Most of today's applications of cryptography ask for authentic data rather than secret data. A practically very important problem is therefore how to protect data and communication against an active attacker (and noise).

**Main related problems to deal with are:**

1. **User identification (authentication):** How can a person prove his (her) identity?
2. **Message authentication:** Can tools be provided to decide, for the recipient, that the message is from the person who is 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 it can be implemented on smart cards - they are essentially credit cards equipped with a chip that can perform arithmetical operations and communications.

**E-commerce:** One of the main new application of the cryptographic techniques is to establish secure and convenient manipulation with digital money (e-money), especially for e-commerce.

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 has actually participated in the identification process. In other words that the Prover has been active in the time the confirmative evidence of 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).

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, pose as the Prover and identify himself (as the Prover) to another Verifier;
- A dishonest party that would claim to be the other party has only negligible chance to identify itself successfully;
- Each of the above conditions remains true even if an attacker has observed or has participated in several identification protocols.

*Identification protocols have to satisfy two security conditions:*

1. If one party, say Bob (a verifier), gets a message from the other party, say 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 out that.

## IV054 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  $r$ .

### 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 a Jane ( a new “player”) with Bob, and get a cryptotext  $e_B(w)$ , the one Jane 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  $w$ .

## USER IDENTIFICATION

**Static means of identification:** People can be identified by their attributes (fingerprints), possessions (passports), or knowledge.

**Dynamic means of identification:** Challenge and respond protocols.

Both Alice and Bob share a key  $k$  and a one-way function  $f_k$ .

1. Bob sends Alice a random number or 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.

The process can be repeated to increase probability of a correct identification.

## Message authentication – to be discussed later

**MAC - method** (Message Authentication Code) Alice and Bob share a key  $k$  and an 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.

A PKC will be used with encryption/decryption algorithms  $(e, d)$  and DSS with pairs  $(s, v)$ . Alice and Bob will have their identity strings  $I_A$  and  $I_B$ .

1. Alice chooses a random  $r_A$ , sets  $t = (I_B, r_A)$ , signs  $sig_{s_A}(t)$  and sends  $m_1 = (t, sig_{s_A}(t))$  to Bob.
2. Bob verifies Alice's signature, chooses random  $r_B$  and a random session key  $k$ . He encrypts  $k$  with Alice's public key,  $E_{e_A}(k) = c$ , sets

$$t_1 = (I_A, r_A, r_B, c),$$

signs it with  $sig_{s_B}(t_1)$ . Then he sends  $m_2 = (t_1, sig_{s_B}(t_1))$  to Alice.

3. Alice verifies Bob's signature, and 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  $k$  via

$$D_{d_A}(c) = D_{d_A}(E_{e_A}(k)) = k,$$

sets  $t_2 = (I_B, r_B)$  and signs it with  $sig_{s_A}(t_2)$ . 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 identity and have agreed upon a session key  $k$ .



# DATA AUTHENTICATION

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) and 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 need to be transmitted through a very secure channel.

## IV054 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 versa.

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
- $K$  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 easy to compute verification mapping
$$ver_k: M \times T \rightarrow \{true, false\}$$

Two conditions should be satisfied for such a scheme:

**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  $m$  from  $M$  and  $k$  from  $K$  it is computationally unfeasible, without a knowledge of  $k$ , to find  $c$  from  $T$  such that  $ver_k(m, c) = true$

Let  $C$  be an encryption algorithm that maps  $k$ -bit strings into  $k$ -bit strings.

If a message

$$m = m_1m_2\dots m_l$$

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  $i = 1, \dots, l$

$$y_i = C(y_{i-1} \oplus m_i)$$

and then

$$y_1 \parallel y_2 \parallel \dots \parallel y_l$$

is the encryption of  $m$  and

$$y_l \text{ is MAC for } m.$$

A modification of this method is to use another crypto-algorithm to encrypt the last block  $m_l$ .

Let us have three pairs and in each: a message and its MAC

$$(m_1, c_1), (m_2, c_2), (m_3, c_3)$$

Where  $m_1$  and  $m_3$  have the same length  $k$  and

$$m_2 = m_1 || B || m'_2.$$

and let the length of  $B$  be also  $k$ . The encryption of the block  $B$  within  $m_2$  is  $C(B \oplus c_1)$ .

If we now define

$$B' = B \oplus c_1 \oplus c_3, m_4 = m_3 || B' || m'_2,$$

then, during the encryption of  $m_4$ , we get

$$C(B' \oplus c_3) = C(B \oplus c_1),$$

This implies that MAC's for  $m_4$  and  $m_2$  are the same.

One can therefore forge a new valid pair

$$(m_4, c_2).$$

**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

$$\text{MAC}(m_1, m_2, \dots, m_l) = C_2(C_1(\dots C_1(C_1(m_1) \oplus m_2) \oplus \dots \oplus m_{l-1} \oplus m_l)).$$

Let us assume that the MAC function be implemented by an oracle, and consider an adversary who can send queries to the oracle with a limited total length of  $q$ . If  $m_1, \dots, m_d$  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  $q = \theta n^{1/2}$ , this is approximately  $a = \theta^2/2$  (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.

## IV054 FROM HASH FUNCTIONS TO MAC

So called **HMAC** was published as the internet standard RFC2104.

Let a hash function **h** processes messages by blocks of **b** bytes and produces 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 strings **opad** and **ipad** defined later)

$$h(k \oplus \text{opad} || h(k \oplus \text{ipad} || m)).$$

and truncate the results to its **t** leftmost bytes to get **HMAX<sub>k</sub>(m)**.

In **HMAX** **ipad** (**opad**) consists of **b** bytes equal to 0x36 (0x5c) hexadecimal.

## IV054 SECURITY of HMAC

It can be shown that if

- $h(k \oplus \text{ipad}||m)$  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

$$\text{MAC}_{k_1, k_2}(m) = h(k_2 || h(k_1 || m))$$

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 **MAC** is a secure **MAC** algorithm for messages of arbitrary length.

## IV054 Disadvantage of static user identification schemes

Everybody who knows your password or PIN can impersonate you.

Using so called **zero-knowledge identification schemes**, discussed in the next chapter, you can identify yourself without giving to the identifier the ability to impersonate you.



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-reponse Identification protocol

- (1) Alice chooses a random  $r < n$ , computes  $x = r^2 \pmod{n}$  and sends  $x$  to Bob.
- (2) Bob sends to Alice a random bit (a **challenge**)  $b$ .
- (3) Alice sends Bob (a **response**)  $y = rs^b \pmod{n}$
- (4) Bob identifies the sender as Alice if and only if  $y^2 = xv^b \pmod{n}$ , what 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 square root  $s$  of  $v$  (without revealing  $s$  to Bob).

If protocol is repeated  $t$  times, Alice has a chance  $2^{-t}$  to fool Bob if she does not know  $s$ .

public-key:  $v$

private-key:  $s$  (of Alice) such that  $s^2 = v$ .

### Protocol

- (1) Alice chooses a random  $r < n$ , computes  $x = r^2 \bmod n$  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  $y^2 = xv^b \bmod n$ , 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 reveals  $r$ .
  - 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  $r \in \mathbb{Z}_n^*$ , 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 Alice.
- Alice sends  $r$  to Bob.

# HOW CAN A BAD EVE CHEAT?

Eve can send, to fool Bob, as her commitment, either  $r^2$  for a random  $r$  or  $r^2v^{-1}$

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.

In the following **parallel version of Fiat-Shamir identification scheme** the probability of false identification is decreased.

Choose primes  $p, q$ , compute  $n = pq$ .

Choose quadratic residues  $v_1, \dots, v_k \in QR_n$ .

Compute  $s_1, \dots, s_k$  such that  $s_i = \sqrt{v_i} \pmod n$

**public-key:**  $v_1, \dots, v_k$

**secret-key:**  $s_1, \dots, s_k$  of Alice

(1) Alice chooses a random  $r < n$ , computes  $a = r^2 \pmod n$  and sends  $a$  to Bob.

(2) Bob sends Alice a random  $k$ -bit string  $b_1 \dots b_k$ .

(3) Alice sends to Bob

$$y = r \prod_{i=1}^k s_i^{b_i} \pmod n$$

(4) Bob accepts if and only if

$$y^2 = a \prod_{i=1}^k v_i^{b_i} \pmod n$$

Alice and Bob repeat this protocol  $t$  times, until Bob is convinced that Alice knows  $s_1, \dots, s_k$ .

The chance that Alice fools Bob is  $2^{-kt}$ , a decrease comparing with the chance  $1/2$  of the previous version of the identification scheme.

This is a **practically attractive and computationally efficient** (in time, space + communication) **scheme** which minimizes storage + computations performed by Alice (to be a smart card).

Scheme requires also a trusted authority (TA) which

- (1) **chooses**: a large prime  $p < 2^{512}$ ,  
 a large prime  $q$  dividing  $p-1$  and  $q \leq 2^{140}$ ,  
 an  $\alpha \in \mathbb{Z}_p^*$  of order  $q$ ,  
 a security parameter  $t$  such that  $2^t < q$ ,  
 $p, q, \alpha, t$  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 string  $ID(Alice)$  which contains identification information.

2. Alice chooses a secret random  $0 \leq a \leq q-1$  and computes

$$v = \alpha^{-a} \bmod p$$

and sends  $v$  to the TA.

3. TA generates signature

$$s = sig_{TA}(ID(Alice), v)$$

and sends to Alice the certificate  $C(Alice) = (ID(Alice), v, s)$

## IV054 Schnorr identification scheme

1. Alice chooses a random  $0 \leq k < q$  and computes

$$\gamma = \alpha^k \bmod p.$$

2. Alice sends her certificate  $C(\text{Alice}) = (\text{ID}(\text{Alice}), v, s)$  and  $\gamma$  to Bob.

3. Bob verifies the signature of the TA by checking that

$$\text{ver}_{\text{TA}}(\text{ID}(\text{Alice}), v, s) = \text{true}.$$

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

$$y = (k + ar) \bmod q.$$

6. Bob verifies that

$$\gamma \equiv \alpha^y v^r \bmod p$$

This way Alice shows her identity to Bob. Indeed,

$$\begin{aligned} \alpha^y v^r &\equiv \alpha^{k+ar} \alpha^{-ar} \bmod p \\ &\equiv \alpha^k \bmod p \\ &\equiv \gamma \bmod p. \end{aligned}$$

**Total storage:** 512 bits for  $\text{ID}(\text{Alice})$ , 512 bits for  $v$ , 320 bits for  $s$  (if DSS is used), total - 1344 bits.

**Total communication:** Alice  $\rightarrow$  Bob 1996 bits,  
Bob  $\rightarrow$  Alice 40 bits.

## IV054 Okamoto identification scheme

The disadvantage of the Schnorr identification scheme is that there is no proof of its security. For the modification of the Schnorr identification scheme presented below, for Okamoto identification scheme, a proof of security exists.

**Basic setting:** To set up the scheme the TA chooses:

- a large prime  $p \leq 2^{512}$ ,
- a large prime  $q \geq 2^{140}$  dividing  $p - 1$ ;
- two elements  $\alpha_1, \alpha_2 \in \mathbb{Z}_p^*$  of order  $q$ .

TA makes public  $p, q, \alpha_1, \alpha_2$  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 an identification string  $ID(Alice)$ .
- Alice secretly and randomly chooses  $0 \leq a_1, a_2 \leq q - 1$  and sends to TA

$$v = \alpha_1^{-a_1} \alpha_2^{-a_2} \bmod p.$$

- TA generates a signature  $s = \text{sig}_{TA}(ID(Alice), v)$  and sends to Alice the certificate  $C(Alice) = (ID(Alice), v, s)$ .



## IV054 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  $\alpha_1, \alpha_2 \in \mathbb{Z}_p^*$  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.

$$v = \alpha_1^{-a_1} \alpha_2^{-a_2} \bmod p.$$

- TA generates a signature  $s = sig_{TA}(ID(Alice), v)$  and sends to Alice the certificate

$$C(Alice) = (ID(Alice), v, s).$$

## IV054 Okamoto identification scheme

### Okamoto identification scheme

- Alice chooses random  $0 \leq k_1, k_2 \leq q-1$  and computes

$$\gamma = \alpha_1^{k_1} \alpha_2^{k_2} \pmod{p}.$$

- Alice sends to Bob her certificate  $(\text{ID}(\text{Alice}), v, s)$  and  $\gamma$ .

- Bob verifies the signature of TA by checking that

$$\text{ver}_{TA}(\text{ID}(\text{Alice}), v, s) = \text{true}.$$

- Bob chooses a random  $1 \leq r \leq 2^t$  and sends it to Alice.

- Alice sends to Bob

$$y_1 = (k_1 + a_1 r) \pmod{q}; \quad y_2 = (k_2 + a_2 r) \pmod{q}.$$

- Bob verifies

$$\gamma \equiv \alpha_1^{y_1} \alpha_2^{y_2} v^r \pmod{p}$$

## IV054 Authentication codes

They provide methods of ensuring integrity of messages - that a message has not been tampered/changed, and that 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:

- 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  $t = t'$  Bob accepts the message as authentic.

## IV054 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 a new 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  $|K| \times |M|$  **authentication matrice** will tabulate all authenticated 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.

## IV054 Example

Let  $M = T = Z_3$ ,  $K = Z_3 \times Z_3$ .

For  $(i, j) \in K$  and  $w \in M$ , let  $t_{ij}(w) = (iw + j) \bmod 3$ .

The matrix **key** x **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  $P_i = 1/3$ .

**Substitution attack:** By checking the table one can see that if Mallot observes an authenticated messages  $(w, t)$ , 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  $P_s = 1/3$ .

**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

$$payoff(w, t) = \Pr(t = a_{k_0}(w)) \quad (4)$$

$$= \sum_{\{k \in K \mid a_k(w) = t\}} \Pr_K(k) \quad (5)$$

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

$$P_i = \max \{payoff(w, t), \mid w \in M, t \in A\}.$$

**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

$$payoff(w', t', w, t) = \Pr(t' = a_{k_0}(w') \mid t = a_{k_0}(w)) \quad (6)$$

$$= \frac{\Pr(t' = a_{k_0}(w') \cap t = a_{k_0}(w))}{\Pr(t = a_{k_0}(w))} \quad (7)$$

$$= \frac{\sum_{\{k \in K \mid a_k(w) = t, a_k(w') = t'\}} \Pr_K(k)}{payoff(w, t)} \quad (8)$$

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  $w'$ .

Since Mallot wants to maximize his chance of deceiving Bob, he needs to compute

$$p_{w,t} = \max \{ \text{payoff}(w', t, w, t) \mid w' \in M, w \neq w', t \in A \}.$$

$p_{w,t}$  therefore denotes the probability that Mallot can deceive Bob with a substitution in the case  $(w, t)$  is the message observed.

If  $Pr_{Ma}(w, t)$  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

$$Pr_{Ma}(w, t) = Pr_M(w) Pr_K(t|w) = Pr_M(w) \times \text{payoff}(w, t).$$

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.

## IV054 Orthogonal arrays

**Definition** An orthogonal array  $OA(n, k, \lambda)$  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  $\lambda$  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  $OA(n, k, \lambda)$ . Then there is an authentication code with  $|M| = k$ ,  $|A| = n$ ,  $|K| = \lambda n^2$  and  $P_i = P_s = 1/n$ .

**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



## IV054 Construction and bounds for OAs

In an orthogonal array  $OA(n, k, \lambda)$

- $n$  determines the number of authenticators (security of the code);
- $k$  is the number of messages the code can accommodate;
- $\lambda$  relates to the number of keys -  $\lambda n^2$ .

The following holds for orthogonal arrays.

- If  $p$  is prime, then  $OA(p, p, 1)$  exists.
- Suppose there exists an  $OA(n, k, \lambda)$ . 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  $OA(p, (p^d-1)/(p-1), p^{d-2})$ .
- Let us have an authentication code with  $|A| = n$  and  $P_i = P_s = 1/n$ . 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  $P_k(k) = 1/n^2$  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.

## IV054 Secret sharing between two parties

A moderator distributes a binary-string secret  $s$ , between two parties  $P_1$  and  $P_2$  by choosing a random binary string  $b$ , of the same length as  $s$ , and

- by sending  $b$  to  $P_1$  and
- by sending  $s \oplus b$  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

$$b \oplus (s \oplus b) = s.$$

## IV054 Threshold secret sharing schemes

Secret sharing schemes distribute a “secret” among several users in such a way that only predefined sets of users 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 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  $n$  participants,  $P = \{P_i \mid 1 \leq i \leq n\}$ , in such a way that any  $t$ , or more, participants can compute the value  $S$ , but no group of  $t-1$ , or less, participants can compute  $S$ .

Secret  $S$  is chosen by a “dealer”  $D \notin P$ .

It is assumed that the dealer “distributes” the secret to participants secretly and in such a way that no participant knows shares of other participants.

## IV054 Shamir's $(n, t)$ -threshold scheme

### Initial phase:

Dealer  $D$  chooses a prime  $p$ ,  $n$  distinct  $x_i$ ,  $1 \leq i \leq n$  and  $D$  gives randomly chosen values  $x_i$  to the user  $P_i$ .

The values  $x_i$  are then public.

**Share distribution:** Suppose  $D$  wants to share a secret  $S \in Z_p$  among the users.  $D$  randomly chooses  $t-1$  elements of  $Z_p$ ,  $a_1, \dots, a_{t-1}$ .

For  $1 \leq i \leq n$ ,  $D$  computes the "shares"  $y_i = a(x_i)$ ,

where

$$a(x) = S + \sum_{j=1}^{t-1} a_j x^j \pmod{p}.$$

For  $1 \leq i \leq n$ ,  $D$  sends the share  $y_i$  to the participant  $P_i$ .

**Secret cumulation:** Let participants  $P_{i_1}, \dots, P_{i_t}$  want to determine secret  $S$ . Since  $a(x)$  has degree  $t-1$ ,  $a(x)$  has the form

$$a(x) = a_0 + a_1 x + \dots + a_{t-1} x^{t-1},$$

and coefficients  $a_i$  can be determined from  $t$  equations  $a(x_{ij}) = y_{ij}$ , 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

$$S = a_0.$$

## IV054 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  $S$  be a set  $\{(x_i, f(x_i)) \mid x_i \in Z_p, i=1, \dots, t, x_i \neq x_j \text{ if } i \neq j\}$ . For any  $Q \subseteq S$ , let  $P_Q = \{g \in Z_p[x] \mid \deg(g) = t-1, g(x) = y \text{ for all } (x, y) \in Q\}$ . 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  $P$ .
- If  $Q$  is a proper subset of  $S$  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  $P = \{(x_i, f(x_i)) \mid i=1, \dots, t, x_i \neq x_j, i \neq j\}$ . Then

$$f(x) = \sum_{i=1}^t f(x_i) \prod_{1 \leq j \leq t, j \neq i} \frac{x - x_j}{x_i - x_j}.$$

## IV054 Shamir's $(n,t)$ -threshold scheme - summary

To distribute  $n$  shares of a secret  $S$  among users  $P_1, \dots, P_n$  a trusted authority  $TA$  proceeds as follows:

- $TA$  chooses a prime  $p > \max\{S, n\}$  and sets  $a_0 = S$ .
- $TA$  selects randomly  $a_1, \dots, a_{t-1} \in \mathbb{Z}_p$  and creates 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 user  $P_i$  in a secure way.

Any group  $J$  of  $t$  or more users 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  $a_0 \in \mathbb{Z}_p$  is equally likely to be the secret.

A serious limitation of the **threshold secret sharing schemes** is that all groups of users with the same number of users have the same access to secret. Practical situations usually require that some (sets of) users are more important than others.

Let  $P$  be a set of users. To deal with above situation such concepts as **authorized set of user** and **access structure** are used.

An **authorized set of users**  $A \subseteq P$  is a set of users who can together construct the secret.

An **unauthorized set of users**  $U \subseteq P$  is a set of users who alone cannot learn anything about the secret.

Let  $P$  be a set of users. 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 protocols increase security of a secret information by sharing it between several subjects.
- Some secret sharing scheme are such that they work even in case some participants 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 participant  $P_i$  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. 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  $g = h^{(p-1)/q} \bmod p$ , are public.

As in Shamir's scheme, the dealer assigns to each participant  $P_i$  a specific  $x_i$  from  $\{1, \dots, q-1\}$  and generates a random polynomial

$$f(x) = \sum_{j=0}^{k-1} a_j x^j \bmod q \quad (1)$$

such that  $f(0) = s$  and sends to each  $P_i$  value  $y_i = f(x_i)$ . In addition, using a broadcasting scheme, the dealer sends to each  $P_i$  all values  $v_j = g^{aj} \bmod p$ .

Each  $P_i$  verifies that

$$g^{y_i} = \prod_{j=0}^{k-1} (v_j)^{x_I^j} \pmod{p}. \quad (1)$$

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 satisfies (1), the dealer is considered as not reliable.

One can easily verify that if the dealer works correctly, then all relations (1) hold

## IV054 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**.

### Basic requirements for e-commerce system:

**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 of e-money.

(**S**ecure **E**lectronic **T**ransaction) protocol was created to standardize the exchange of credit card information. Development of **SET** initiated in 1996 the credit card companies MasterCard and Visa.

## IV054 DUAL SIGNATURE PROTOCOL

We present a protocol to solve the following security and privacy problem in e-commerce: shoppers **banks** should not know what **cardholders** are ordering and **shops** should not learn credit cards numbers.

Participants of our e-commerce protocol: a **bank**, a **cardholder**, a **shop**

The **cardholder** uses the following information:

- **GSO** - **Goods and Service Order** (cardholder's name, shop's name, items being ordered, their quantity,...)
- **PI** - **Payment instructions** (shop's name, card number, total price,...)

**Protocol** uses a public hash function  $h$ .

**RSA cryptosystem** is used and

- $e_C$ ,  $e_S$  and  $e_B$  are public keys of **cardholder**, **shop**, **bank** and
- $d_C$ ,  $d_S$  and  $d_B$  are their secret keys.

## IV054 CARDHOLDER and SHOP ACTIONS

A **cardholder** performs the following procedure--**GSO**-goods and service 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.
3. Computes  $HPO = h(HEPI || HEGSO)$  - Hash values of the **P**ayment **O**rder.
4. Signs **HPO** by computing "Dual Signature"  $DS = d_c(HPO)$ .
5. Sends  $e_s(GSO)$ ,  $DS$ ,  $HEPI$ , and  $e_B(PI)$  to **shop**.

**Shop** does the following: (payment instructions)

- Calculates  $h(e_s(GSO)) = HEGSO$ ;
- Calculates  $h(HEPI || HEGSO)$  and  $e_c(DS)$ . If they are equal, **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**.

## IV054 BANK and SHOP ACTIONS

Bank has received  $HEPI$ ,  $HEGSO$ ,  $e_B(PI)$ , and  $DS$  and performs the following actions.

1. Computes  $h(e_B(PI))$  - what should be equal to  $HEPI$ .
2. Computes  $h(h(e_B(PI)) || HEGSO)$  what 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 fulfils basic requirements concerning security, privacy and integrity.

## IV054 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 should 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 system of e-money have been created that satisfy all or at least some of the above requirements.

## IV054 BLIND SIGNATURES - applications

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 have 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 Shnorr blind signature protocol described on the next slide.

### Basic setting

Bank chooses large primes  $p, q \mid (p-1)$  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  $x \in \{0, \dots, p-1\}$ .

Public information:  $(p, q, g, y = g^x)$ .



1. Shnorr's simplified identification protocol 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  $c = h(m \parallel a)$ , 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  $r' \in \{0, \dots, q-1\}$ .
- Bob chooses random  $u, v, w \in \{0, \dots, q-1\}$ ,  $u \neq 0$ , computes  $a = a'^u g^v y^w$ ,  $c = h(m \parallel a)$ ,  $c' = (c - w)u^{-1}$  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  $\sigma(m) = (c, b)$  of  $m$ .

Verification condition for the blind signature:  $c = h(m \parallel g^b y^c)$ .

Both  $(a, c, b)$  and  $(a', c', b')$  are valid transcripts.