Part I

Identification, authentication, secret sharing and e-commerce

EXAMS

- December: 21.12.2018 at 9.30 in B410
- January: 08.01.2019 at 12.00 in B410
  15.01.2019 at 12.00 in B410
  22.01.2018 at 12.00 in B410

WISDOM

Keep in mind that a cryptosystem is as secure as its weakest part - security does not add up!

CHAPTER 9: AUTHENTICATION, SECRET SHARING and e-COMMERCE

CONTENTS I. - USER IDENTIFICATION and MESSAGE AUTHENTICATION/INTEGRITY

Most of today's cryptographic applications ask for identification of communicating parties, and/or for data integrity/authentication during communication, rather than for secrecy of transferred data.

CONTENTS I. - USER IDENTIFICATION and MESSAGE AUTHENTICATION/INTEGRITY

Most of today's cryptographic applications ask for identification of communicating parties, and/or for data integrity/authentication during communication, rather than for secrecy of transferred data.

Main related problems to deal with are:

1. **User identification (authentication):** How can a person/computer prove her/his identity?

CONTENTS I. - USER IDENTIFICATION and MESSAGE AUTHENTICATION/INTEGRITY

Most of today's cryptographic applications ask for identification of communicating parties, and/or for data integrity/authentication during communication, rather than for secrecy of transferred 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 received is indeed from the person who was supposed to send it?

CONTENTS I. - USER IDENTIFICATION and MESSAGE AUTHENTICATION/INTEGRITY

Most of today's cryptographic applications ask for identification of communicating parties, and/or for data integrity/authentication during communication, rather than for secrecy of transferred 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 received is indeed from the person who was supposed to send it?
3. **Message integrity:** Can tools be provided to decide for the recipient whether or not the message was changed on the fly?

CONTENTS I. - USER IDENTIFICATION and MESSAGE AUTHENTICATION/INTEGRITY

Most of today's cryptographic applications ask for identification of communicating parties, and/or for data integrity/authentication during communication, rather than for secrecy of transferred 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 received is indeed from the person who was supposed to send it?
3. **Message integrity:** 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.

CONTENTS I. - USER IDENTIFICATION and MESSAGE AUTHENTICATION/INTEGRITY

Most of today's cryptographic applications ask for identification of communicating parties, and/or for data integrity/authentication during communication, rather than for secrecy of transferred 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 received is indeed from the person who was supposed to send it?
3. **Message integrity:** 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.

With all of the above problems we will deal in the first part of this chapter.

MORE FORMALLY and MORE GENERALLY

- **Authentication** is, in particular, the act of confirmation the identity of a communicating entity (a person or a computer), and, in general, the act of confirmation the truth of an attribute, datum or entity.

MORE FORMALLY and MORE GENERALLY

- **Authentication** is, in particular, the act of confirmation the identity of a communicating entity (a person or a computer), and, in general, the act of confirmation the truth of an attribute, datum or entity.
- **Data integrity** refers to maintaining and ensuring the accuracy and consistency of data over its entire life cycle - the accuracy, validity and correctness of data should be ensured from hardware failures, software errors and human errors.

CONTENTS II. - SECRET SHARING and E-COMMERCE

CONTENTS II. - SECRET SHARING and E-COMMERCE

Secret sharing problem is the problem how to share a "secret" among a group of users in such a way that only well specified subsets of them can determine the secret.

CONTENTS II. - SECRET SHARING and E-COMMERCE

Secret sharing problem is the problem how to share a "secret" among a group of users in such a way that only well specified subsets of them can determine the secret.

Secret sharing schemes are ideal, for example, for storing information that is highly sensitive and important. For example, for encryption keys.

CONTENTS II. - SECRET SHARING and E-COMMERCE

Secret sharing problem is the problem how to share a "secret" among a group of users in such a way that only well specified subsets of them can determine the secret.

Secret sharing schemes are ideal, for example, for storing information that is highly sensitive and important. For example, for encryption keys.

Secret sharing protocols/schemes are another often used cryptographic primitives, with a variety of applications, we will deal with in second part of this chapter.

CONTENTS II. - SECRET SHARING and E-COMMERCE

Secret sharing problem is the problem how to share a "secret" among a group of users in such a way that only well specified subsets of them can determine the secret.

Secret sharing schemes are ideal, for example, for storing information that is highly sensitive and important. For example, for encryption keys.

Secret sharing protocols/schemes are another often used cryptographic primitives, with a variety of applications, we will deal with in second part of this chapter.

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.

CONTENTS II. - SECRET SHARING and E-COMMERCE

Secret sharing problem is the problem how to share a "secret" among a group of users in such a way that only well specified subsets of them can determine the secret.

Secret sharing schemes are ideal, for example, for storing information that is highly sensitive and important. For example, for encryption keys.

Secret sharing protocols/schemes are another often used cryptographic primitives, with a variety of applications, we will deal with in second part of this chapter.

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.

An example how e-commerce can be realized, in a simplified setting, will be shown at the end of this chapter.

USER IDENTIFICATION (AUTHENTICATION)

User identification (authentication) is a process at which one party (often referred to as a Prover or as Alice), convinces another party (often referred to as a Verifier or as 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 as Alice), convinces another party (often referred to as a Verifier or as Bob) of Prover's identity.

Namely, that the Prover (Alice) convinces the other party that she has indeed participated (or is participating) in the identification process.

USER IDENTIFICATION (AUTHENTICATION)

User identification (authentication) is a process at which one party (often referred to as a Prover or as Alice), convinces another party (often referred to as a Verifier or as Bob) of Prover's identity.

Namely, that the Prover (Alice) convinces the other party that she has indeed participated (or is participating) in the identification process.

In other words that the Prover has been herself active in proving her identity in the time the 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 as Alice), convinces another party (often referred to as a Verifier or as Bob) of Prover's identity.

Namely, that the Prover (Alice) convinces the other party that she has indeed participated (or is participating) in the identification process.

In other words that the Prover has been herself active in proving her identity in the time the 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.

USER IDENTIFICATION (AUTHENTICATION)

User identification (authentication) is a process at which one party (often referred to as a Prover or as Alice), convinces another party (often referred to as a Verifier or as Bob) of Prover's identity.

Namely, that the Prover (Alice) convinces the other party that she has indeed participated (or is participating) in the identification process.

In other words that the Prover has been herself active in proving her identity in the time the 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:

OBJECTIVES of IDENTIFICATIONS

User identification process has to satisfy the following objectives:

- **The Verifier will accept Prover's identity if both parties are honest;**

OBJECTIVES of IDENTIFICATIONS

User identification process has to satisfy the following objectives:

- **The Verifier will accept Prover's identity if both parties are honest;**
- The Verifier cannot later, after participating in a successful identification, learn how to act as the Prover and to identify himself (as the Prover) to another verifier;

OBJECTIVES of IDENTIFICATIONS

User identification process has to satisfy the following objectives:

- **The Verifier will accept Prover's identity if both parties are honest;**
- The Verifier cannot later, after participating in a successful identification, learn how to act as the Prover and to identify himself (as the Prover) to another verifier;
- **A third party (called attacker here), say E, following the identification process of the Prover to the Verifier, has only a negligible chance to identify itself to someone else successfully as the Prover;**

OBJECTIVES of IDENTIFICATIONS

User identification process has to satisfy the following objectives:

- **The Verifier will accept Prover's identity if both parties are honest;**
- The Verifier cannot later, after participating in a successful identification, learn how to act as the Prover and to identify himself (as the Prover) to another verifier;
- **A third party (called attacker here), say E, following the identification process of the Prover to the Verifier, has only a negligible chance to identify itself to someone else successfully as the Prover;**
- Each of the above conditions should remain valid even if an attacker has observed, or has even participated in, several identification processes of the same party.

USER IDENTIFICATION PROTOCOLS

Identification protocols have to satisfy two security conditions:

USER 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, that claims to be Alice (a Prover), then Bob should be able to verify that the sender was indeed Alice.

USER 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, that claims to be Alice (a Prover), then Bob should be able to verify that the sender was indeed Alice.
2. There should be 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

IDENTIFICATION SYSTEM BASED on a PKC

- Alice chooses a random r and sends $e_B(r)$ to Bob.

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.

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 potential misuse of the above system

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 potential misuse of the above system

We show that (any non-honest) Alice could misuse the above identification scheme.

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 potential 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.

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 potential 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 always 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

IDENTIFICATION SYSTEM BASED on a PKC - a better version

- Alice chooses a random r and sends $e_B(r)$ to Bob.

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.

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),

USER IDENTIFICATION

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

USER IDENTIFICATION

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

USER IDENTIFICATION

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

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** (of a key or of a method).

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

Example: Let both Alice and Bob share a key k and a one-way function f_k.

USER IDENTIFICATION

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

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

Example: Let 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.

USER IDENTIFICATION

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

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

Example: Let 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 to Bob** $PI = f_k(RAND)$.

USER IDENTIFICATION

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

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

Example: Let 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 to Bob** $PI = f_k(RAND)$.
3. **If Bob gets** PI, then he verifies whether $PI = f_k(RAND)$.

USER IDENTIFICATION

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

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

Example: Let 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 to 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

USER IDENTIFICATION

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

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

Example: Let 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 to 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 Alice 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** (of a key or of a method).

Dynamic means of identification: Challenge and respond protocols.

Example: Let 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 to 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 Alice 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** (of a key or of a method).

Dynamic means of identification: Challenge and respond protocols.

Example: Let 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 to 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 Alice 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

USER IDENTIFICATION

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

Dynamic means of identification: Challenge and respond protocols.

Example: Let 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 to 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 Alice 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 (**M**essage **A**uthentication **C**ode) Let Alice and Bob share a key **k** and an encoding algorithm A_k

USER IDENTIFICATION

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

Dynamic means of identification: Challenge and respond protocols.

Example: Let 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 to 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 Alice 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 (**M**essage **A**uthentication **C**ode) Let Alice and Bob share a key **k** and an encoding algorithm A_k

1. **To communicate a message m**, Alice sends a pair$(m, A_k(m))$ – $\{A_k(m)$ is said to be **MAC**$\}$.
2. If Bob gets (m', MAC), then he computes $A_k(m')$ and compares it with **MAC**.

CHALLENGE-RESPONSE PROTOCOLS - A SPECIFICATION

CHALLENGE-RESPONSE PROTOCOLS - A SPECIFICATION

In a **challenge-response identification protocol** a party A proves its identity to a party B by demonstrating knowledge of a secret/method known to be associated with A only,

CHALLENGE-RESPONSE PROTOCOLS - A SPECIFICATION

In a **challenge-response identification protocol** a party A proves its identity to a party B by demonstrating knowledge of a secret/method known to be associated with A only, without revealing the secret/method itself to B.

CHALLENGE-RESPONSE PROTOCOLS - A SPECIFICATION

In a **challenge-response identification protocol** a party A proves its identity to a party B by demonstrating knowledge of a secret/method known to be associated with A only, without revealing the secret/method itself to B.

Structure of challenge-response protocols:

In a **challenge-response identification protocol** a party A proves its identity to a party B by demonstrating knowledge of a secret/method known to be associated with A only, without revealing the secret/method itself to B.

Structure of challenge-response protocols:

1. Commitment (to a secret).

In a **challenge-response identification protocol** a party A proves its identity to a party B by demonstrating knowledge of a secret/method known to be associated with A only, without revealing the secret/method itself to B.

Structure of challenge-response protocols:

1. Commitment (to a secret).
2. Challenge.

CHALLENGE-RESPONSE PROTOCOLS - A SPECIFICATION

In a **challenge-response identification protocol** a party A proves its identity to a party B by demonstrating knowledge of a secret/method known to be associated with A only, without revealing the secret/method itself to B.

Structure of challenge-response protocols:

1. Commitment (to a secret).
2. Challenge.
3. Response.

CHALLENGE-RESPONSE PROTOCOLS - A SPECIFICATION

In a **challenge-response identification protocol** a party A proves its identity to a party B by demonstrating knowledge of a secret/method known to be associated with A only, without revealing the secret/method itself to B.

Structure of challenge-response protocols:

1. Commitment (to a secret).
2. Challenge.
3. Response.
4. Verification (of the response).

THREE-WAY AUTHENTICATION and also KEY-AGREEMENT I

In this protocol a PKC will be used with encryption/decryption algorithms (e_U, d_U), for each user U, and a DSS with signing/verification algorithms(sig_U, ver_U).

THREE-WAY AUTHENTICATION and also KEY-AGREEMENT I

In this protocol a PKC will be used with encryption/decryption algorithms (e_U, d_U), for each user U, and a DSS with signing/verification algorithms(sig_U, ver_U). In addition, Alice and Bob will have their, public, identification strings I_A and I_B.

THREE-WAY AUTHENTICATION and also KEY-AGREEMENT I

In this protocol a PKC will be used with encryption/decryption algorithms (e_U, d_U), for each user U, and a DSS with signing/verification algorithms(sig_U, ver_U). In addition, Alice and Bob will have their, public, identification strings I_A and I_B.

1. Alice chooses a random integer r_A,

THREE-WAY AUTHENTICATION and also KEY-AGREEMENT I

In this protocol a PKC will be used with encryption/decryption algorithms (e_U, d_U), for each user U, and a DSS with signing/verification algorithms(sig_U, ver_U). In addition, Alice and Bob will have their, public, identification strings I_A and I_B.

1. Alice chooses a random integer r_A, sets $\mathrm{t} = (I_B,\ r_A)$,

THREE-WAY AUTHENTICATION and also KEY-AGREEMENT I

In this protocol a PKC will be used with encryption/decryption algorithms (e_U, d_U), for each user U, and a DSS with signing/verification algorithms(sig_U, ver_U). In addition, Alice and Bob will have their, public, identification strings I_A and I_B.

1. Alice chooses a random integer r_A, sets $t = (I_B,\ r_A)$, signs it as $sig_A(I_A, t)$

THREE-WAY AUTHENTICATION and also KEY-AGREEMENT I

In this protocol a PKC will be used with encryption/decryption algorithms (e_U, d_U), for each user U, and a DSS with signing/verification algorithms(sig_U, ver_U). In addition, Alice and Bob will have their, public, identification strings I_A and I_B.

1. Alice chooses a random integer r_A, sets $\mathrm{t} = (I_B,\ r_A)$, signs it as $sig_A(I_A, t)$ and sends $m_1 = (t, sig_A(I_A, t))$ to Bob.

THREE-WAY AUTHENTICATION and also KEY-AGREEMENT I

In this protocol a PKC will be used with encryption/decryption algorithms (e_U, d_U), for each user U, and a DSS with signing/verification algorithms(sig_U, ver_U). In addition, Alice and Bob will have their, public, identification strings I_A and I_B.

1. Alice chooses a random integer r_A, sets $\mathrm{t} = (I_B,\ r_A)$, signs it as $sig_A(I_A, t)$ and sends $m_1 = (t, sig_A(I_A, t))$ to Bob.
2. Bob verifies Alice's signature,

THREE-WAY AUTHENTICATION and also KEY-AGREEMENT I

In this protocol a PKC will be used with encryption/decryption algorithms (e_U, d_U), for each user U, and a DSS with signing/verification algorithms(sig_U, ver_U). In addition, Alice and Bob will have their, public, identification strings I_A and I_B.

1. Alice chooses a random integer r_A, sets $\mathrm{t} = (I_B,\ r_A)$, signs it as $sig_A(I_A, t)$ and sends $m_1 = (t, sig_A(I_A, t))$ 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

In this protocol a PKC will be used with encryption/decryption algorithms (e_U, d_U), for each user U, and a DSS with signing/verification algorithms(sig_U, ver_U). In addition, Alice and Bob will have their, public, identification strings I_A and I_B.

1. Alice chooses a random integer r_A, sets $t = (I_B,\ r_A)$, signs it as $sig_A(I_A, t)$ and sends $m_1 = (t, sig_A(I_A, t))$ 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 $e_A(k) = c$, sets

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

   and signs it as $sig_B(t_1)$.

THREE-WAY AUTHENTICATION and also KEY-AGREEMENT I

In this protocol a PKC will be used with encryption/decryption algorithms (e_U, d_U), for each user U, and a DSS with signing/verification algorithms(sig_U, ver_U). In addition, Alice and Bob will have their, public, identification strings I_A and I_B.

1. Alice chooses a random integer r_A, sets $\mathrm{t} = (I_B,\ r_A)$, signs it as $sig_A(I_A, t)$ and sends $m_1 = (t, sig_A(I_A, t))$ 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 $e_A(k) = c$, sets

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

and signs it as $sig_B(t_1)$. Then he sends $m_2 = (t_1, sig_B(t_1))$ 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.

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.

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 also gets the session key k via computation

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

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.

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 also gets the session key k via computation
   $$D_{d_A}(c) = D_{d_A}(E_{e_A}(k)) = k,$$
   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.

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 also gets the session key k via computation

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

   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 others 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 unreliable (and/or insecure) channels.

The goal of data authentication schemes (protocols) is to handle the case that data are sent through unreliable (and/or insecure) channels.

By creating a so-called Message Authentication Code (MAC) and sending this MAC, together with the 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 unreliable (and/or insecure) channels.

By creating a so-called Message Authentication Code (MAC) and sending this MAC, together with the 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.

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 for that message.

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 for that message.

A scheme (M, T, K) for a 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 for that message.

A scheme (M, T, K) for a 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 $k \in K$ there is a single and easy to compute authentication mapping

$$auth_k : \{0,1\}^* \times M \rightarrow T$$

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 for that message.

A scheme (M, T, K) for a 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 $k \in K$ 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 for that message.

A scheme (M, T, K) for a 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 $k \in K$ there is a single and easy to compute authentication mapping

$$auth_k : \{0,1\}^* \times M \to T$$

- and a single and easy to compute verification mapping

$$ver_k : M \times T \to \{true, false\}$$

such that the following two conditions should be satisfied:

Correctness: For each $m \in M$ and $k \in K$ the following holds: $ver_k(m, c) = true$ if there exists an $r \in \{0,1\}^*$ such that $c = auth_k(r, m)$

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 for that message.

A scheme (M, T, K) for a 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 $k \in K$ 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 $m \in M$ and $k \in K$ the following holds: $ver_k(m, c) = true$ if there exists an $r \in \{0,1\}^*$ 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 determine $t \in T$ 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

$$m = m_1 m_2 \ldots m_l$$

is divided into blocks of length k,

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 the length k, and performs the following computations, for i = 1, . . . ,l

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

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 the length k, and performs the following computations, for $i = 1, \dots, l$

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

In such a case

$$y_1 \| y_2 \| \dots \| y_l$$

is the encryption of m and

y_l can 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.

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 so called **challenge and response (or challenge-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 so called **challenge and response (or challenge-response) protocols**.

Basic idea of such protocols:

- Let Alice be known to have a capability 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 so called **challenge and response (or challenge-response) protocols**.

Basic idea of such protocols:

- Let Alice be known to have a capability to solve some hard problem P.
- Bob challenges a party claimed to be Alice 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 so called **challenge and response (or challenge-response) protocols**.

Basic idea of such protocols:

- Let Alice be known to have a capability to solve some hard problem P.
- Bob challenges a party claimed to be Alice by asking her to solve a particular instance of the P problem.
- If the party succeeds, Bob intends to believe that 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 so called **challenge and response (or challenge-response) protocols**.

Basic idea of such protocols:

- Let Alice be known to have a capability to solve some hard problem P.
- Bob challenges a party claimed to be Alice by asking her to solve a particular instance of the P problem.
- If the party succeeds, Bob intends to believe that 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.

CHALLENGE-RESPONSE PROTOCOLS - A GENERAL SPECIFICATION

CHALLENGE-RESPONSE PROTOCOLS - A GENERAL SPECIFICATION

In a **challenge-response identification protocol** a party A proves its identity to a party B by demonstrating knowledge of a secret/method known to be associated with A only,

CHALLENGE-RESPONSE PROTOCOLS - A GENERAL SPECIFICATION

In a **challenge-response identification protocol** a party A proves its identity to a party B by demonstrating knowledge of a secret/method known to be associated with A only, without revealing the secret/method itself to B.

CHALLENGE-RESPONSE PROTOCOLS - A GENERAL SPECIFICATION

In a **challenge-response identification protocol** a party A proves its identity to a party B by demonstrating knowledge of a secret/method known to be associated with A only, without revealing the secret/method itself to B.

Structure of challenge-response protocols:

CHALLENGE-RESPONSE PROTOCOLS - A GENERAL SPECIFICATION

In a **challenge-response identification protocol** a party A proves its identity to a party B by demonstrating knowledge of a secret/method known to be associated with A only, without revealing the secret/method itself to B.

Structure of challenge-response protocols:

1. Commitment (to a secret).

CHALLENGE-RESPONSE PROTOCOLS - A GENERAL SPECIFICATION

In a **challenge-response identification protocol** a party A proves its identity to a party B by demonstrating knowledge of a secret/method known to be associated with A only, without revealing the secret/method itself to B.

Structure of challenge-response protocols:

1. Commitment (to a secret).
2. Challenge.

CHALLENGE-RESPONSE PROTOCOLS - A GENERAL SPECIFICATION

In a **challenge-response identification protocol** a party A proves its identity to a party B by demonstrating knowledge of a secret/method known to be associated with A only, without revealing the secret/method itself to B.

Structure of challenge-response protocols:

1. Commitment (to a secret).
2. Challenge.
3. Response.

CHALLENGE-RESPONSE PROTOCOLS - A GENERAL SPECIFICATION

In a **challenge-response identification protocol** a party A proves its identity to a party B by demonstrating knowledge of a secret/method known to be associated with A only, without revealing the secret/method itself to B.

Structure of challenge-response protocols:

1. Commitment (to a secret).
2. Challenge.
3. Response.
4. Verification (of the response).

BASIC 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 $s^2 \equiv v \pmod{n}$.

BASIC 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 $s^2 \equiv v \pmod{n}$.

public-key: v

private-key: s (that Alice knows, but not Bob)

BASIC 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 $s^2 \equiv v \pmod{n}$.

public-key: v

private-key: s (that Alice knows, but not Bob)

BASIC 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 $s^2 \equiv v \pmod n$.

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, as her **commitment**, to Bob.

BASIC 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 $s^2 \equiv v \pmod n$.

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, as her **commitment**, to Bob.
2. Bob sends to Alice a random bit (as his challenge) b.

BASIC 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 $s^2 \equiv v \pmod{n}$.

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, as her **commitment**, to Bob.
2. Bob sends to Alice a random bit (as his challenge) b.
3. Alice sends Bob (as her response) $y = rs^b \mod n$

BASIC 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 $s^2 \equiv v \pmod{n}$.

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, as her **commitment**, to Bob.
2. Bob sends to Alice a random bit (as his challenge) b.
3. Alice sends Bob (as her response) $y = rs^b \mod n$
4. Bob identifies the sender as Alice if and only if, **verification**, $y^2 = xv^b \mod n$ holds, which is taken as a proof that the sender knows square roots of x and of v.

BASIC 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 $s^2 \equiv v \pmod{n}$.

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, as her **commitment**, to Bob.
2. Bob sends to Alice a random bit (as his challenge) b.
3. Alice sends Bob (as her response) $y = rs^b \mod n$
4. Bob identifies the sender as Alice if and only if, **verification**, $y^2 = xv^b \mod n$ holds, 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

BASIC 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 $s^2 \equiv v \pmod{n}$.

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, as her **commitment**, to Bob.
2. Bob sends to Alice a random bit (as his challenge) b.
3. Alice sends Bob (as her response) $y = rs^b \mod n$
4. Bob identifies the sender as Alice if and only if, **verification**, $y^2 = xv^b \mod n$ holds, 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.

BASIC 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 $s^2 \equiv v \pmod{n}$.

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, as her **commitment**, to Bob.
2. Bob sends to Alice a random bit (as his challenge) b.
3. Alice sends Bob (as her response) $y = rs^b \mod n$
4. Bob identifies the sender as Alice if and only if, **verification**, $y^2 = xv^b \mod n$ holds, 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

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 (a **commitment**) to Bob.

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 (a **commitment**) to Bob.
2. Bob sends to Alice a random bit b **(a challenge).**

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 (a **commitment**) to Bob.
2. Bob sends to Alice a random bit b **(a challenge).**
3. Alice sends to Bob (a response) $y = rs^b$.

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 (a **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 (a **verification**) if and only if $y^2 = xv^b \mod n$, proving that Alice knows a square root of **x**.

Analysis

ANALYSIS of Fiat-Shamir IDENTIFICATION II

Analysis

1. The first message is **a commitment** by Alice that she knows a square root of x.

ANALYSIS of Fiat-Shamir IDENTIFICATION II

Analysis

1. The first message is **a commitment** by Alice that she knows a square root of x.
2. The second message is **a challenge** by Bob.

ANALYSIS of Fiat-Shamir IDENTIFICATION II

Analysis

1. The first message is **a commitment** by Alice that she knows a 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.

ANALYSIS of Fiat-Shamir IDENTIFICATION II

Analysis

1. The first message is **a commitment** by Alice that she knows a 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".

Analysis

1. The first message is **a commitment** by Alice that she knows a 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 a 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.

HOW CAN BAD EVE CHEAT?

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

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 $b = 1$.

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 $b = 0$.

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, (computationally efficient, in time, space + communication) identification 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, (computationally efficient, in time, space + communication) identification 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.

THE SCHNORR IDENTIFICATION SCHEME – SETTING

This is a practically attractive, (computationally efficient, in time, space + communication) identification 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, (computationally efficient, in time, space + communication) identification 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.

THE SCHNORR IDENTIFICATION SCHEME – SETTING

This is a practically attractive, (computationally efficient, in time, space + communication) identification 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.

THE SCHNORR IDENTIFICATION SCHEME – SETTING

This is a practically attractive, (computationally efficient, in time, space + communication) identification 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

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

   and sends to Alice as her certificate: 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 C (Alice) = (ID(Alice), v, s) and also γ.
3. Bob verifies the signature of TA by checking that
$$ver_{\text{TA}}(ID(Alice), v, s) = true.$$

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

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
$$y = (k + ar) \mod q.$$
6. Bob verifies that
$$\gamma \equiv \alpha^y v^r \mod p$$

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
$$y = (k + ar) \mod q.$$
6. Bob verifies that
$$\gamma \equiv \alpha^y v^r \mod p$$
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 → Bob – 1996 (= 1344+512+140) bits,
Bob → 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 so called 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 so called 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 so called 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 $s = sig_{\text{TA}}(ID(Alice), v)$ and sends to Alice the certificate

$$\text{C (Alice)} = (\text{ID(Alice), v, s}).$$

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

DATA (MESSAGE) INTEGRITY
and
AUTHENTICATION

DATA INTEGRITY and AUTHENTICATION PROBLEMS

DATA INTEGRITY and AUTHENTICATION PROBLEMS

- One of the main features of the current information processing era is that it becomes more and more a **(big) data-driven era** - society is accumulating enormous amounts of data and has big problems with its reliable and efficient storing, transmission and processing.

DATA INTEGRITY and AUTHENTICATION PROBLEMS

- One of the main features of the current information processing era is that it becomes more and more a **(big) data-driven era** - society is accumulating enormous amounts of data and has big problems with its reliable and efficient storing, transmission and processing.
- In general, **data integrity** refers to maintaining and assuring the **accuracy and consistency of data over their whole real life cycle** and becomes a very important feature of database systems.

DATA INTEGRITY and AUTHENTICATION PROBLEMS

- One of the main features of the current information processing era is that it becomes more and more a **(big) data-driven era** - society is accumulating enormous amounts of data and has big problems with its reliable and efficient storing, transmission and processing.
- In general, **data integrity** refers to maintaining and assuring the **accuracy and consistency of data over their whole real life cycle** and becomes a very important feature of database systems.
- The goal is to ensure accuracy, validity and correctness of data - a protection from hardware, software and human errors.

DATA INTEGRITY and AUTHENTICATION PROBLEMS

- One of the main features of the current information processing era is that it becomes more and more a **(big) data-driven era** - society is accumulating enormous amounts of data and has big problems with its reliable and efficient storing, transmission and processing.
- In general, **data integrity** refers to maintaining and assuring the **accuracy and consistency of data over their whole real life cycle** and becomes a very important feature of database systems.
- The goal is to ensure accuracy, validity and correctness of data - a protection from hardware, software and human errors.
- In database systems, data integrity is normally enforced by a series of so called **integrity constrains/rules**.

DATA INTEGRITY and AUTHENTICATION PROBLEMS

- One of the main features of the current information processing era is that it becomes more and more a **(big) data-driven era** - society is accumulating enormous amounts of data and has big problems with its reliable and efficient storing, transmission and processing.
- In general, **data integrity** refers to maintaining and assuring the **accuracy and consistency of data over their whole real life cycle** and becomes a very important feature of database systems.
- The goal is to ensure accuracy, validity and correctness of data - a protection from hardware, software and human errors.
- In database systems, data integrity is normally enforced by a series of so called **integrity constrains/rules**.
- Closely related to data integrity problems is the problem of authentication of data at their transmissions.

DATA INTEGRITY and AUTHENTICATION PROBLEMS

- One of the main features of the current information processing era is that it becomes more and more a **(big) data-driven era** - society is accumulating enormous amounts of data and has big problems with its reliable and efficient storing, transmission and processing.
- In general, **data integrity** refers to maintaining and assuring the **accuracy and consistency of data over their whole real life cycle** and becomes a very important feature of database systems.
- The goal is to ensure accuracy, validity and correctness of data - a protection from hardware, software and human errors.
- In database systems, data integrity is normally enforced by a series of so called **integrity constrains/rules**.
- Closely related to data integrity problems is the problem of authentication of data at their transmissions.
- With the use of cryptographic techniques to deal with data authentication problem we deal briefly in the next.

AUTHENTICATION CODES

They provide methods to ensure authentication of data/messages – that a message has not been tampered/changed, and that the message originated with the presumed sender.

AUTHENTICATION CODES

They provide methods to ensure authentication of data/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.

AUTHENTICATION CODES

They provide methods to ensure authentication of data/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 authentication of data/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 $k \in K$

Transmission process

AUTHENTICATION CODES

They provide methods to ensure authentication of data/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.

AUTHENTICATION CODES

They provide methods to ensure authentication of data/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 $k \in K$

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 $t = a_k(w)$.

AUTHENTICATION CODES

They provide methods to ensure authentication of data/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 $k \in K$

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.

THE AUTHENTICATION MATRIX - EXAMPLE

Let $M = T = Z_3$, $K = Z_3 \times Z_3 - -Z_3 = \{0, 1, 2\}$.
For $(i, j) \in K$ and $w \in M$, let $t_{ij}(w) = (iw + j) \mod 3$.

THE AUTHENTICATION MATRIX - EXAMPLE

Let $M = T = Z_3,\ K = Z_3 \times Z_3 - -Z_3 = \{0, 1, 2\}$.
For $(i, j) \in K$ and $w \in M$, let $t_{ij}(w) = (iw + j) \mod 3$.
The matrix key $\times$ message of authentication tags has now 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

THE AUTHENTICATION MATRIX - EXAMPLE

Let $M = T = Z_3,\ K = Z_3 \times Z_3 - -Z_3 = \{0, 1, 2\}$.
For $(i, j) \in K$ and $w \in M$, let $t_{ij}(w) = (iw + j) \mod 3$.
The matrix key $\times$ message of authentication tags has now 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: Let us assume that Mallot picks a message w and tries to guess the correct authentication tag.

THE AUTHENTICATION MATRIX - EXAMPLE

Let $M = T = Z_3,\ K = Z_3 \times Z_3 - -Z_3 = \{0, 1, 2\}$.
For $(i, j) \in K$ and $w \in M$, let $t_{ij}(w) = (iw + j) \mod 3$.
The matrix key $\times$ message of authentication tags has now 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: Let us assume that Mallot picks a message w and tries to guess the correct authentication tag.
Problem is that for each message w and each tag a there are exactly three keys k such that $t_k(w) = a$. Hence $P_i = \frac{1}{3}$.

THE AUTHENTICATION MATRIX - EXAMPLE

Let $M = T = Z_3$, $K = Z_3 \times Z_3 - -Z_3 = \{0, 1, 2\}$.
For $(i, j) \in K$ and $w \in M$, let $t_{ij}(w) = (iw + j) \mod 3$.
The matrix key $\times$ message of authentication tags has now 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: Let us assume that Mallot picks a message w and tries to guess the correct authentication tag.
Problem is that 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, a), then there are exactly three possibilities for the key that was used.
Moreover, for each choice (w', a'), w $\neq$ w', there is exactly one of the three possible keys for (w',a') 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 $|M| = k, |T| = n, |K| = \lambda n^2$ 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/tags (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/tags (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/tags (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/tags (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/tags (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.

COMMENTS on ORTHOGONAL ARRAYS

- Orthogonal arrays are a very important concept of recreational mathematics, combinatorial mathematics, coding theory.
- They were introduced by Rao in 1946.
- One of the non-trivial questions is for which parameters one can construct the corresponding Orthogonal array.
- There is a library of more than 200 Orthogonal arrays.

SECRET SHARING

Secret sharing refers to methods for distribution of "secrets" amongst a group of users (usually called players),

SECRET SHARING

Secret sharing refers to methods for distribution of "secrets" amongst a group of users (usually called players), using so called "shares of the secret",

SECRET SHARING

Secret sharing refers to methods for distribution of "secrets" amongst a group of users (usually called players), using so called "shares of the secret", in such a way that only so called "eligible groups of players" can determine the secret by their cooperation using their shares only.

SECRET SHARING

Secret sharing refers to methods for distribution of "secrets" amongst a group of users (usually called players), using so called "shares of the secret", in such a way that only so called "eligible groups of players" can determine the secret by their cooperation using their shares only.

Secret sharing problem was discovered, as an important cryptographic primitive, independently by Adi Shamir and George Blakeley in 1979 and they also constructed first secret sharing protocols.

SECRET SHARING

Secret sharing refers to methods for distribution of "secrets" amongst a group of users (usually called players), using so called "shares of the secret", in such a way that only so called "eligible groups of players" can determine the secret by their cooperation using their shares only.

Secret sharing problem was discovered, as an important cryptographic primitive, independently by Adi Shamir and George Blakeley in 1979 and they also constructed first secret sharing protocols.

For example, secret sharing is used as cryptographic primitive in several protocols for secure multiparty computation.

SECRET SHARING - PROBLEM

In some 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 subsets of parties can determine the secret - if members of the parties cooperate.

SECRET SHARING - PROBLEM

In some 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 subsets of parties can determine the secret - if members of the parties cooperate.

For example, in some cases one can increase security of confidential information, say a secret key, by sharing it between several parties.

SECRET SHARING - PROBLEM

In some 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 subsets of parties can determine the secret - if members of the parties cooperate.

For example, in some 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, or more, of parties can create S, but no $t - 1$, or less, of parties will have the slightest idea how to get the secret.

BASIC IDEA of the (n,t) THRESHOLD SECRET SHARING

In order to distribute a secret (number) S among n parties, a dealer creates a degree $t-1$ random polynomial p such that p(0)=S

BASIC IDEA of the (n,t) THRESHOLD SECRET SHARING

In order to distribute a secret (number) S among n parties, a dealer creates a degree $t-1$ random polynomial p such that p(0)=S and distributes to parties, as their "shares" of the secret, – values of p in separate points of p.

BASIC IDEA of the (n,t) THRESHOLD SECRET SHARING

In order to distribute a secret (number) S among n parties, a dealer creates a degree $t-1$ random polynomial p such that p(0)=S and distributes to parties, as their "shares" of the secret, – values of p in separate points of p.

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 the **slightest idea** about S.

SECRET SHARING between TWO PARTIES

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 as follows:

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 as follows: **He chooses a random binary string b, of the same length as s, and**

- **sends b (as a "share" of s), to P_1**

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 as follows: **He chooses a random binary string b, of the same length as s, and**

- **sends b (as a "share" of s), to P_1 and**
- **sends $s \oplus b$ (as another share of s), to P_2.**

A dealer creates shares of a binary-string secret s and distributes them between two parties P_1 and P_2 as follows: **He chooses a random binary string b, of the same length as s, and**

- **sends b (as a "share" of s), to P_1 and**
- **sends $s \oplus b$ (as another share of s), to P_2.**

This way, none of the parties P_1 and P_2 alone has a slightest idea about s,

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 as follows: **He chooses a random binary string b, of the same**

length as s, and

- **sends b (as a "share" of s), to P_1 and**
- **sends $s \oplus b$ (as another share of s), 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

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

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 as follows: **He chooses a random binary string b, of the same length as s, and**

- **sends b (as a "share" of s), to P_1 and**
- **sends $s \oplus b$ (as another share of s), 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 - FORMALITIES

THRESHOLD SECRET SHARING SCHEMES - FORMALITIES

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 - FORMALITIES

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.

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 - FORMALITIES

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.

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.

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" $D \notin P$.

THRESHOLD SECRET SHARING SCHEMES - FORMALITIES

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.

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.

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.

THE CASE $n = t$

THE CASE $n = t$

Such a case is easy to deal with.

THE CASE $n = t$

Such a case is easy to deal with.

In the case of an m bit secret S,

each but one of n parties is assigned a different m bit random number

THE CASE $n = t$

Such a case is easy to deal with.

In the case of an m bit secret S,

each but one of n parties is assigned a different m bit random number

and the last participant gets, as his share

THE CASE $n = t$

Such a case is easy to deal with.

In the case of an m bit secret S,

each but one of n parties is assigned a different m bit random number

and the last participant gets, as his share $X \oplus S$, where X is xor of all remaining random shares.

By xoring all shares the secret S can be obtained.

BASIC PROPERTIES of SECURE SECRET SHARING SCHEMES

- All shares have to be "as large as the secret" in an (n, t) secret sharing scheme.

BASIC PROPERTIES of SECURE SECRET SHARING SCHEMES

- All shares have to be "as large as the secret" in an (n, t) secret sharing scheme.

  Indeed, any share SH_i has to have the property that no group of $t - 1$ of the remaining shares contains any information about the secret, but adding the share SH_i, the secret can be obtained.

BASIC PROPERTIES of SECURE SECRET SHARING SCHEMES

- All shares have to be "as large as the secret" in an (n, t) secret sharing scheme.

  Indeed, any share SH_i has to have the property that no group of $t-1$ of the remaining shares contains any information about the secret, but adding the share SH_i, the secret can be obtained.

  Therefore:

BASIC PROPERTIES of SECURE SECRET SHARING SCHEMES

- All shares have to be "as large as the secret" in an (n, t) secret sharing scheme.

  Indeed, any share SH_i has to have the property that no group of $t-1$ of the remaining shares contains any information about the secret, but adding the share SH_i, the secret can be obtained.

  Therefore: (1) No share can contain "some information about secret";

BASIC PROPERTIES of SECURE SECRET SHARING SCHEMES

- All shares have to be "as large as the secret" in an (n, t) secret sharing scheme.

  Indeed, any share SH_i has to have the property that no group of $t - 1$ of the remaining shares contains any information about the secret, but adding the share SH_i, the secret can be obtained.

  Therefore: (1) No share can contain "some information about secret"; (2) but also each share has to contain "all information about the secret" -

BASIC PROPERTIES of SECURE SECRET SHARING SCHEMES

- All shares have to be "as large as the secret" in an (n, t) secret sharing scheme.

  Indeed, any share SH_i has to have the property that no group of $t - 1$ of the remaining shares contains any information about the secret, but adding the share SH_i, the secret can be obtained.

  Therefore: (1) No share can contain "some information about secret"; (2) but also each share has to contain "all information about the secret" - both in some sense.

BASIC PROPERTIES of SECURE SECRET SHARING SCHEMES

- All shares have to be "as large as the secret" in an (n, t) secret sharing scheme.

  Indeed, any share SH_i has to have the property that no group of $t-1$ of the remaining shares contains any information about the secret, but adding the share SH_i, the secret can be obtained.

  Therefore: (1) No share can contain "some information about secret"; (2) but also each share has to contain "all information about the secret" - both in some sense.
- All secure secret sharing schemes have to use random elements.

Shamir's (n,t)-THRESHOLD SCHEME

Initial phase:

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

Shamir's (n,t)-THRESHOLD SCHEME

Initial phase:

Dealer D chooses a prime p, n randomly chosen 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 chosen 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.

Shamir's (n,t)-THRESHOLD SCHEME

Initial phase:
Dealer D chooses a prime p, n randomly chosen 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. (1) **D** randomly chooses, and keeps secret, t - 1 elements of Z_p, $a_1, \ldots, a_{t-1}$.

Shamir's (n,t)-THRESHOLD SCHEME

Initial phase:
Dealer D chooses a prime p, n randomly chosen 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. (1) **D** randomly chooses, and keeps secret, t - 1 elements of Z_p, $a_1, \ldots, a_{t-1}$.
(b) For $1 \leq i \leq n$, the dealer D computes "shares" $y_i = a(x_i)$, where

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

Shamir's (n,t)-THRESHOLD SCHEME

Initial phase:
Dealer D chooses a prime p, n randomly chosen 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. (1) **D** randomly chooses, and keeps secret, t - 1 elements of Z_p, $a_1, \ldots, a_{t-1}$.
(b) For $1 \leq i \leq n$, the dealer D computes "shares" $y_i = a(x_i)$, where

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

(3) Finally, D 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 chosen 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. (1) **D** randomly chooses, and keeps secret, t - 1 elements of Z_p, $a_1, \ldots, a_{t-1}$.

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

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

(3) Finally, D sends the share y_i to the party P_i, $1 \leq i \leq n$ and keeps coefficients a_i secret.

Secret accumulation:

Shamir's (n,t)-THRESHOLD SCHEME

Initial phase:
Dealer D chooses a prime p, n randomly chosen 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. (1) **D** randomly chooses, and keeps secret, t - 1 elements of Z_p, $a_1, \ldots, a_{t-1}$.
(b) For $1 \leq i \leq n$, the dealer D computes "shares" $y_i = a(x_i)$, where

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

(3) Finally, D 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 the secret S.

Shamir's (n,t)-THRESHOLD SCHEME

Initial phase:
Dealer D chooses a prime p, n randomly chosen 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. (1) **D** randomly chooses, and keeps secret, t - 1 elements of Z_p, $a_1, \ldots, a_{t-1}$.
(b) For $1 \leq i \leq n$, the dealer D computes "shares" $y_i = a(x_i)$, where

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

(3) Finally, D 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 the secret S. Since, unknown to them, polynomial a(x) has degree t-1 they know that it has, in general, 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.

Shamir's (n,t)-THRESHOLD SCHEME

Initial phase:
Dealer D chooses a prime p, n randomly chosen 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. (1) **D** randomly chooses, and keeps secret, t - 1 elements of Z_p, $a_1, \ldots, a_{t-1}$.
(b) For $1 \leq i \leq n$, the dealer D computes "shares" $y_i = a(x_i)$, where

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

(3) Finally, D 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 the secret S. Since, unknown to them, polynomial a(x) has degree t-1 they know that it has, in general, 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 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 chosen 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. (1) **D** randomly chooses, and keeps secret, t - 1 elements of Z_p, $a_1, \ldots, a_{t-1}$.
(b) For $1 \leq i \leq n$, the dealer D computes "shares" $y_i = a(x_i)$, where

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

(3) Finally, D 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 the secret S. Since, unknown to them, polynomial a(x) has degree t-1 they know that it has, in general, 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 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.

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

$$\Omega = \{(x_i, f(x_i)) \,|\, x_i \in Z_p, i = 1, \ldots, t, x_i \neq x_j \text{ if } i \neq j\}$$

For any $Q \subseteq \Omega$, let $P_Q = \{g \in Z_p[x] | deg(g) \leq t-1, g(x) = y$ for all (x,y) $\in Q\}$. Then it holds:

- $P_\Omega = \{f(x)\}$, 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

$$\Omega = \{(x_i, f(x_i)) \,|\, x_i \in Z_p, i = 1, \ldots, t, x_i \neq x_j \text{ if } i \neq j\}$$

For any $Q \subseteq \Omega$, let $P_Q = \{g \in Z_p[x] | deg(g) \leq t-1, g(x) = y$ for all (x,y) $\in Q\}$. Then it holds:

- $P_\Omega = \{f(x)\}$, 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 $P_1, \ldots, P_n$ 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 $a_1, \ldots, a_{t-1} \in Z_p$ and creates the polynomial $f(x) = \sum_{i=0}^{t-1} a_i x^i$.
- TA computes $s_i = f(i), i = 1, \ldots, n$ 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.

PROPERTIES of SHAMIR's SECRET (n, t) SHARING SCHEMES

- **Security:** The scheme is information theoretically secure.
- **Minimality:** The size of each share does not exceed the size of the secret.
- **Dynamicity:** Shares can be replaced by another ones without affecting other shares.
- **Flexibility:** Parties can obtain different number of shares according to their importance (within an organization they are in).

ORTHOGONAL ARRAYS BASED SHARING SCHEME

ORTHOGONAL ARRAYS BASED SHARING SCHEME

General form of orthogonal arrays: An $t-(n,k,\lambda)$ orthogonal array for $t \leq k$ is a $\lambda n^t \times k$ array, whose entries are from a set X of n points such that in every subset of t columns of the array, every t-tuple of points of X appears in exactly λ rows.

ORTHOGONAL ARRAYS BASED SHARING SCHEME

General form of orthogonal arrays: An $t-(n,k,\lambda)$ orthogonal array for $t \leq k$ is a $\lambda n^t \times k$ array, whose entries are from a set X of n points such that in every subset of t columns of the array, every t-tuple of points of X appears in exactly λ rows.

A $t-(n,n+1,1)$ orthogonal array may be used to construct a perfect (n,t) threshold secret sharing scheme, in the following way:

ORTHOGONAL ARRAYS BASED SHARING SCHEME

General form of orthogonal arrays: An $t-(n,k,\lambda)$ orthogonal array for $t \leq k$ is a $\lambda n^t \times k$ array, whose entries are from a set X of n points such that in every subset of t columns of the array, every t-tuple of points of X appears in exactly λ rows.

A $t-(n,n+1,1)$ orthogonal array may be used to construct a perfect (n,t) threshold secret sharing scheme, in the following way:

Let A be an $t-(v,n+1,1)$ orthogonal array.

ORTHOGONAL ARRAYS BASED SHARING SCHEME

General form of orthogonal arrays: An $t-(n,k,\lambda)$ orthogonal array for $t \leq k$ is a $\lambda n^t \times k$ array, whose entries are from a set X of n points such that in every subset of t columns of the array, every t-tuple of points of X appears in exactly λ rows.

A $t-(n,n+1,1)$ orthogonal array may be used to construct a perfect (n,t) threshold secret sharing scheme, in the following way:

Let A be an $t-(v,n+1,1)$ orthogonal array. The first n columns will be used to provide shares to the parties,

ORTHOGONAL ARRAYS BASED SHARING SCHEME

General form of orthogonal arrays: An $t-(n,k,\lambda)$ orthogonal array for $t \leq k$ is a $\lambda n^t \times k$ array, whose entries are from a set X of n points such that in every subset of t columns of the array, every t-tuple of points of X appears in exactly λ rows.

A $t-(n,n+1,1)$ orthogonal array may be used to construct a perfect (n,t) threshold secret sharing scheme, in the following way:

Let A be an $t-(v,n+1,1)$ orthogonal array. The first n columns will be used to provide shares to the parties, while the last column represents the secret to be shared.

ORTHOGONAL ARRAYS BASED SHARING SCHEME

General form of orthogonal arrays: An $t-(n,k,\lambda)$ orthogonal array for $t \leq k$ is a $\lambda n^t \times k$ array, whose entries are from a set X of n points such that in every subset of t columns of the array, every t-tuple of points of X appears in exactly λ rows.

A $t-(n,n+1,1)$ orthogonal array may be used to construct a perfect (n,t) threshold secret sharing scheme, in the following way:

Let A be an $t-(v,n+1,1)$ orthogonal array. The first n columns will be used to provide shares to the parties, while the last column represents the secret to be shared.If the dealer wishes to share a secret S only the rows of A where the last entry is S are used in the scheme.

ORTHOGONAL ARRAYS BASED SHARING SCHEME

General form of orthogonal arrays: An $t-(n,k,\lambda)$ orthogonal array for $t \leq k$ is a $\lambda n^t \times k$ array, whose entries are from a set X of n points such that in every subset of t columns of the array, every t-tuple of points of X appears in exactly λ rows.

A $t-(n,n+1,1)$ orthogonal array may be used to construct a perfect (n,t) threshold secret sharing scheme, in the following way:

Let A be an $t-(v,n+1,1)$ orthogonal array. The first n columns will be used to provide shares to the parties, while the last column represents the secret to be shared.If the dealer wishes to share a secret S only the rows of A where the last entry is S are used in the scheme. The dealer then randomly selects one of these rows and sends out to the party P_i the entry in this raw and in the column i as the share.

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 an **authorized set of users** of P and **access structures** are used.

An 'authorized set of parties $A \subseteq P$ is a set of parties who should be able, when cooperating, to construct the secret.

An **unauthorized set of parties** $U \subseteq P$ is a set of parties who alone should not be able to 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 an **authorized set of users** of P and **access structures** are used.

An 'authorized set of parties $A \subseteq P$ is a set of parties who should be able, when cooperating, to construct the secret.

An **unauthorized set of parties** $U \subseteq P$ is a set of parties who alone should not be able to learn anything about the secret.

Let P be a set of parties. The **access structure** $\Gamma \subseteq 2^P$ is a set of subsets of parties 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.

EXAMPLE of an ACCESS STRUCTURE

An access structure for the set of players

$$P = \{P_1, P_2, P_3, P_4, P_5\}$$

is the set of subsets of P that contains sets

$$\{P_2, P_5\}, \quad \{P_1, P_4\} \quad \{P_1, P_2, P_3\}$$

and all their supersets.

SECRET SHARING SCHEME with VERIFICATION

- Secret sharing protocols increase security of a secret information by sharing it between several parties.

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 party P_i is capable to verify correctness of his/her share s_i

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 party 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 other parties about its correctness

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 party 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 other parties about its correctness

In general, a player might lie about his own share, in order to gain information about other shares. Secret sharing schemes with verification allow players to be certain that none of the other players is lying about his share.

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 $g = h^{\frac{p-1}{q}} \mod p$, will be 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 $g = h^{\frac{p-1}{q}} \mod p$, will be 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 $\mathrm{f(0) = S}$ and sends to each P_i the value $y_i = f(x_i)$.

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 $g = h^{\frac{p-1}{q}} \mod p$, will be 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 the value $y_i = f(x_i)$. In addition, using a broadcasting scheme, the dealer sends to each P_i all values $v_j = g^{a_j} \mod p$.

Feldman's (n,k)-PROTOCOL - continuation

Each P_i verifies that

$$g^{y_i} = \prod_{j=0}^{k-1} (v_j)^{x_i^j} \mod p \qquad (1)$$

Feldman's (n,k)-PROTOCOL - continuation

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.

Feldman's (n,k)-PROTOCOL - continuation

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.

Feldman's (n,k)-PROTOCOL - continuation

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.

Feldman's (n,k)-PROTOCOL - continuation

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.
Observe that $(v_j)^{x_i^j} = g^{a_j x_i^j}$ and therefore

$$g^{y_i} = \prod_{j=0}^{k-1} (v_j)^{x_i^j} \mod p = g^{f(x_i)}$$

(1)

SECRET SHARING using CHINESE REMAINDER THEOREM

There are at least two threshold secret sharing schemes in which shares are generated by reduction of a secret S modulo some integers m_i and the secret is essentially recovered by solving a system of linear congruences using the Chinese remainder Theorem.

SECRET SHARING using CHINESE REMAINDER THEOREM

There are at least two threshold secret sharing schemes in which shares are generated by reduction of a secret S modulo some integers m_i and the secret is essentially recovered by solving a system of linear congruences using the Chinese remainder Theorem.

Basic idea for (n, t) secret sharing scheme: Choose n relatively prime integers $m_1 < m_2 < \ldots < m_n$, and a secret

$$\prod_{i=`n-t+2}^{n} m_i < S < \prod_{i=1}^{t} m_i.$$

SECRET SHARING using CHINESE REMAINDER THEOREM

There are at least two threshold secret sharing schemes in which shares are generated by reduction of a secret S modulo some integers m_i and the secret is essentially recovered by solving a system of linear congruences using the Chinese remainder Theorem.

Basic idea for (n, t) secret sharing scheme: Choose n relatively prime integers $m_1 < m_2 < \ldots < m_n$, and a secret

$$\prod_{i=`n-t+2}^{n} m_i < S < \prod_{i=1}^{t} m_i.$$

i-th share will be $s_i = S \bmod m_i$ Recovery of the secret S from the shares $s_{i_1}, s_{i_2}, \ldots s_{i_t}$ is done by solving system of equations

$$S \equiv s_{i_j} \bmod m_{i_j}, j = 1, 2, \ldots t$$

Observe that the above condition for S implies that S is smaller than the product of any choice t of m's, but, at the same time, greater than any choice of $t-1$ of them.

Blakley's SECRET SHARING SCHEME

Blakley's SECRET SHARING SCHEME

This is a secret sharing scheme based on the following facts:

Blakley's SECRET SHARING SCHEME

This is a secret sharing scheme based on the following facts:

Blakley's SECRET SHARING SCHEME

This is a secret sharing scheme based on the following facts:

- Two nonparallel lines in the same plane intersect at exactly one point.
- Three nonparallel planes in space intersect in exactly one point.
- In general any n nonparallel $(n-1)$-dimensional hyperplanes intersect in exactly one point.

The secret can be therefore encoded as any single coordinate of the point of the intersection of n nonparallel $(n-1)$-dimensional hyperplanes.

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 transparencies are overlaid.

E-COMMERCE

Very important is to ensure security of e-money transactions needed for e-commerce.

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 SYSTEMS

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 of e-money.

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: banks, shoppers/cardholders, shops

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: banks, shoppers/cardholders, shops

The cardholder will use the following information:

- GSO – Goods and Services Order (cardholder's name, shop's name, items being ordered, their quantity,...)

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: banks, shoppers/cardholders, shops

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,...)

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: banks, shoppers/cardholders, shops

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: banks, shoppers/cardholders, shops

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 the 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 a GSO-goods and services order

1. Computes $HEGSO = h(e_S(GSO))$ – the hash value of the encryption of GSO.

CARDHOLDER and SHOP ACTIONS

A cardholder performs the following procedure – to create a GSO-goods and services order

1. Computes $HEGSO = h(e_S(GSO))$ – the 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.

CARDHOLDER and SHOP ACTIONS

A cardholder performs the following procedure – to create a GSO-goods and services order

1. Computes $HEGSO = h(e_S(GSO))$ – the 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 a GSO-goods and services order

1. Computes $HEGSO = h(e_S(GSO))$ – the 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 a GSO-goods and services order

1. Computes $HEGSO = h(e_S(GSO))$ – the 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.

CARDHOLDER and SHOP ACTIONS

A cardholder performs the following procedure – to create a GSO-goods and services order

1. Computes $HEGSO = h(e_S(GSO))$ – the 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 a GSO-goods and services order

1. Computes $HEGSO = h(e_S(GSO))$ – the 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 a GSO-goods and services order

1. Computes $HEGSO = h(e_S(GSO))$ – the 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 a GSO-goods and services order

1. Computes $HEGSO = h(e_S(GSO))$ – the 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 a GSO-goods and services order

1. Computes $HEGSO = h(e_S(GSO))$ – the 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.

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.

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.

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 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-moneys have to be signed by a Bank.

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-moneys 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.

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-moneys 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 Schnorr blind signature protocol described on the next slide.

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-moneys 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 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 $x \in \{0, \dots, p-1\}$.
Public information: $(p, q, g, y = g^x)$.

BLIND SIGNATURES – protocols

BLIND SIGNATURES – protocols

1. Schnorr's simplified identification scheme in which Bank proves its identity by proving that it knows 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 $r \in \{0, \ldots, q-1\}$ and send $a = g^r$ to Bob.

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 $r \in \{0, \ldots, q-1\}$ 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}.

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 $r \in \{0, \ldots, q-1\}$ 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}.

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 $r \in \{0, \ldots, q-1\}$ 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 $y = g^x$}

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 $r \in \{0, \ldots, q-1\}$ 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 the message to be signed.

   Signature: (c, b); Verification rule: $a = g^b y^c$; Transcript: (a, c, b).

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 $r \in \{0, \ldots, q-1\}$ 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 the message to be signed.

   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: $c = h(m \| g^b y^c)$.

APPENDIX

SOME BASIC CONCEPTS Of APPLIED CRYPTOGRAPHY

In applied cryptography literature the following concepts are often used:

- random string - a string obtained by tossing coins.
- nonce - a number that is used only once (in a use of a protocol).
- salt - a short random string.
- salting (padding) - attaching a short random string - a salt

A use of such concepts will be illustrated in the next.

DIGITAL CASH TRANSACTIONS

Basic players and procedures:

Bank uses RSA with encryption (decryption) exponent e (d) and modulus n.

Digital money; (m, m^d), where m is unique identification number of a coin, m^d is its bank signature. Bank records all coin identification numbers in a *database of used coins* together with an identification of the money owner.

Blind signatures - blinding To sign a coin m by a bank, customer (Bob) chooses a random r, sends $t = r^e m \pmod{n}$ to bank. the bank signs it and sends $u = t^d$ to Bob. By computing ur^{-1} Bob gets m^d.

E-cash withdraw

- Bob generates 100 sets of 100 unique strings $S_j = \{I_{j_k}\}_{k=1}^{100}$, $1 \leq j \leq 100$, such that each I_{j_k} uniquely identifies Bob.
- Bob splits each I_{j_k} into two pieces $I_{j_k} = (L_{j_k}, R_{j_k})$.
- Bob sends to bank 100 blinded money orders

$$M_j = (100\$, m_j, r_j^e m_j, \{L_{j_k}, R_{j_k}\}_{k=1}^{100}),$$

  where all m_j and r_j are randomly chosen.
- Bank chooses randomly one of 100 money orders, say M_{100}, checks that all remaining ones are for the same amounts, have different m_j and that each $L_{j_k} \oplus R_{j_k}$ identifies Bob. If all is O.K. Bank signs M_j.
- Bob unblinds signature to get ECash coin (m_{100}, m_{100}^d).

E-CASH SPENDING

1. Shop verifies bank's signature by computing $(m_{100}^d)^e = m_{100}$.
2. Shop sends Bob a random binary string $b_1 b_2 \dots b_{100}$ and asks Bob to reveal L_{100_k} if $b_k = 1$ and R_{100_k} if $b_k = 0$ what Bob does, for all k. Afterwards, shop sends the money order to bank together with the chosen binary string b and Bob's responses.
3. Bank checks its used coins database. If m_{100} is not there, bank deposits 100$ into shop's account and m_{100} into its used coins database, together with Bob's identification, and let shop to know that the money order is. O.K. Shop then sends goods to Bob.
4. If m_{100} is in the database of used coins, the money order is rejected. Bank then compares the identity string on false money order with the stored identity string attached to m_{100}. If they are the same, bank knows that shop duplicated the money order. If they differ, then bank knows that the entity who gave it to the shop must have copied it. In case the coin (m_{100}, m_{100}^d) was spent with another shop, then that shop gave Bob another binary string (in step 2). Bank compares corresponding binary strings to find an i, where i-th bits differ. This means that one shop asked Bob to reveal R_i and second L_i. By computing $L_i \oplus R_i$ bank reveals Bob's identity, which can be reported to authorities.