Introduction to cryptographic protocols

Stéphanie Delaune

November 6, 2007

→ Thanks to Véronique Cortier (some of the slides come from her presentation)
Cryptographic protocols

- small programs designed to secure communication (various security goals)
- use cryptographic primitives (e.g. encryption, hash function,
Cryptographic protocols

- small programs designed to secure communication (various security goals)
- use cryptographic primitives (e.g. encryption, hash function,
Security properties (1)

- **Secrecy**: May an intruder learn some secret message between two honest participants?

- **Authentication**: Is the agent Alice really talking to Bob?

- **Fairness**: Alice and Bob want to sign a contract. Alice initiates the protocol. May Bob obtain some advantage?

- **Privacy**: Alice participate to an election. May a participant learn something about the vote of Alice?

- **Non-repudiation**: Alice sends a message to Bob. Alice cannot later deny having sent this message. Bob cannot deny having received the message.

  ...
Security properties: E-voting (2)

**Eligibility:** only legitimate voters can vote, and only once

**Fairness:** no early results can be obtained which could influence the remaining voters

**Individual verifiability:** a voter can verify that her vote was really counted

**Universal verifiability:** the published outcome really is the sum of all the votes
Security properties: E-voting (3)

Privacy: the fact that a particular voted in a particular way is not revealed to anyone

Receipt-freeness: a voter cannot prove that she voted in a certain way (this is important to protect voters from coercion)

Coercion-resistance: same as receipt-freeness, but the coercer interacts with the voter during the protocol, (e.g. by preparing messages)
Cryptographic primitives

Algorithms that are frequently used to build computer security systems. These routines include, but are not limited to, one-way hash functions and encryption functions.
Cryptographic primitives

Algorithms that are frequently used to build computer security systems. These routines include, but are not limited to, **one-way hash functions** and **encryption** functions.

Examples: Caesar encryption, DES, AES, . . .
Security of cryptographic protocols

How cryptographic protocols can be attacked?

Breaking encryption

- Ciphertext-only attack,
- Known-plaintext attack, ...
Logical attack - What is it?

Electronic bank transfer
Logical attack - What is it?

Electronic bank transfer

\[ \{ N \}_\text{pub(client)} \]

\[ N \]
Logical attack - What is it?

Electronic bank transfer

\[ \text{client} \rightarrow \{N\}_\text{pub} \]
Logical attack - What is it?

Electronic bank transfer
Logical attack - What is it?

Electronic bank transfer

\[
\{ N \}^{\text{pub}}(\text{client}) \quad \rightarrow \quad N \quad \leftarrow \quad \{ N \}^{\text{pub}}(\text{client})
\]
Logical attack - What is it?

Electronic bank transfer

Logical attacks

- can be mounted even assuming perfect cryptography, → replay attack, man-in-the-middle attack, ...
- are numerous, see SPORE, Security Protocols Open REpository → http://www.lsv.ens-cachan.fr/spore/
- subtle and hard to detect by “eyeballing” the protocol
Outline of the talk

1. Introduction

2. Cryptographic primitives
   - Symmetric encryption
   - Asymmetric encryption
   - Signature
   - Hash function

3. Cryptographic protocols
   - Toss of a coin by phone
   - A simple key distribution protocol
   - Needham-Schroeder public-key protocol
   - Credit Card payment
   - Zero-knowledge protocols

4. Conclusion
Outline of the talk

1. Introduction

2. Cryptographic primitives
   - Symmetric encryption
   - Asymmetric encryption
   - Signature
   - Hash function

3. Cryptographic protocols
   - Toss of a coin by phone
   - A simple key distribution protocol
   - Needham-Schroeder public-key protocol
   - Credit Card payment
   - Zero-knowledge protocols

4. Conclusion
Symmetric encryption: How does it work?
Caesar cipher

A type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet.
Caesar cipher

A type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet.

Example: A shift of 3.

Plaintext: THE QUICK FOX JUMPS OVER THE LAZY DOG
Ciphertext: WKH TXLFN IRA MXPSV RYHU WKH ODCB GRJ
Caesar cipher

A type of **substitution** cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet.

**Example:** A shift of 3.

Plaintext: THE QUICK FOX JUMPS OVER THE LAZY DOG

Ciphertext: WKH TXLFN IRA MXPSV RYHU WKH ODCB GRJ

→ Breaking encryption: frequency analysis
World War II cryptography

Enigma

- (electro)mechanical cipher machine,
- many substitutions and permutations: designed to defeat basic cryptanalytic techniques by continually changing the substitution alphabet.

Breaking encryption:

- **1945**: Almost all German Enigma traffic could be decrypted within a day or two.
- **Alan Turing** made important contributions to efficient Enigma-breaking.
Perfect cryptography: one-time pad

One-time pad

An encryption algorithm where the plaintext is combined with a random key or "pad" that is as long as the plaintext and used only once.

\[ \{m\}_k = m \oplus k \] where \( \oplus \) is the exclusive or operation
Perfect cryptography: one-time pad

One-time pad

An encryption algorithm where the plaintext is combined with a random key or "pad" that is as long as the plaintext and used only once.

\[ \{m\} = m \oplus k \] where \( \oplus \) is the exclusive or operation.

Example:

\begin{align*}
\text{message} & \quad m = 0101 \\
\text{key} & \quad k = 1100 \\
\text{cipher} & \quad \{m\}_k = 1001
\end{align*}
Perfect cryptography: one-time pad

One-time pad

An encryption algorithm where the plaintext is combined with a random key or "pad" that is as long as the plaintext and used only once.

→ \{m\}_k = m \oplus k \text{ where } \oplus \text{ is the exclusive or operation}

Example:

\begin{align*}
\text{message} & \quad m & = & \quad 0101 \\
\text{key} & \quad k & = & \quad 1100 \\
\text{cipher} & \quad \{m\}_k & = & \quad 1001
\end{align*}

- **Advantages**: perfect secrecy, efficient,
- **Drawbacks**: long random keys that can only be used once,
- **Applications**: « telephone rouge » between U.S. and the Soviet Union (1963), key delivered with a so-called diplomatic bag.
DES - Data Encryption Standard (1)

→ a product cipher developed by IBM (1977)

Modern cryptography

Modern secret key cryptography often uses the same ingredients (permutations and substitutions) to make cryptanalysis almost impossible.
DES - Data Encryption Standard (1)

→ a product cipher developed by IBM (1977)

Modern cryptography

Modern secret key cryptography often uses the same ingredients (permutations and substitutions) to make cryptanalysis almost impossible.

Security:

- the security of this scheme (as most of the symmetric encryption scheme) is not provable,
- DES is now considered to be insecure for many applications → the length of the key is too small (64 bits = 56 + 8)
**Ingredients**

- **P-boxes**: devices that accept a certain length bit string and produce a permutation of the bits.
- **S-boxes**: devices that accept a certain length bit string; the output is the result of applying a substitution cipher to the bit string.
Some others symmetric encryption schemes

Triple DES

Triple DES is formed from the Data Encryption Standard (DES) cipher by using it three times with 2 or 3 different keys.

- a product cipher developed by IBM (1978)
- 3DES is slowly disappearing from use (replaced by AES).
  → One large-scale exception is within the electronic payments industry, e.g. Cash machines.
Some others symmetric encryption schemes

Triple DES

Triple DES is formed from the Data Encryption Standard (DES) cipher by using it three times with 2 or 3 different keys.

- a product cipher developed by IBM (1978)
- 3DES is slowly disappearing from use (replaced by AES).
  → One large-scale exception is within the electronic payments industry, e.g. Cash machines.

AES: Advances Encryption Standard

→ developed by Vincent Rijmen, Joan Daemen (1998)
→ AES is one of the most popular algorithms
To encrypt long messages

Encryption algorithms allow us to encrypt small messages (a block)

Questions?
How can we encrypt long messages?
To encrypt long messages

Encryption algorithms allow us to encrypt small messages (a block)

Questions?

How can we encrypt long messages?

- **ECB**: Electronic CodeBook
- **CBC**: Cipher Block Chaining
Electronic Codebook (ECB) mode encryption
Cipher Block Chaining (CBC) mode encryption
What about security?

A pixel-map version of an image encrypted with
Outline of the talk

1. Introduction

2. Cryptographic primitives
   - Symmetric encryption
   - Asymmetric encryption
   - Signature
   - Hash function

3. Cryptographic protocols
   - Toss of a coin by phone
   - A simple key distribution protocol
   - Needham-Schroeder public-key protocol
   - Credit Card payment
   - Zero-knowledge protocols

4. Conclusion
Asymmetric encryption: How does it work?

Asymmetric encryption

Encryption

Decryption

Public key

Private key
RSA - Rivest, Shamir and Adleman

publicly described in 1977 by Rivest, Shamir, and Adleman at MIT

\[ n = pq \quad d = e^{-1} \mod \phi(n) \]

\( e \) (public key) \quad \( d \) (private key)
RSA - Rivest, Shamir and Adleman

→ publicly described in 1977 by Rivest, Shamir, and Adleman at MIT

<table>
<thead>
<tr>
<th>public</th>
<th>private</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n = pq )</td>
<td>( d = e^{-1} \mod \phi(n) )</td>
</tr>
<tr>
<td>( e ) (public key)</td>
<td>( d ) (private key)</td>
</tr>
</tbody>
</table>

Encryption \( E(m) = m^e \mod n \)

Decryption \( D(c) = c^d \mod n \)
RSA - Rivest, Shamir and Adleman

→ publicly described in 1977 by Rivest, Shamir, and Adleman at MIT

\[
\begin{array}{c|c}
\text{public} & \text{private} \\
\hline
n = pq & d = e^{-1} \mod \phi(n) \\
e \text{ (public key)} & d \text{ (private key)}
\end{array}
\]

Encryption \( E(m) = m^e \mod n \)
Decryption \( D(c) = c^d \mod n \)

\[
m \overset{\text{encrypt}}{\longrightarrow} m^e \mod n \overset{\text{decrypt}}{\rightarrow} (m^e \mod n)^d \mod n = m^{ed} \mod n = m^1 \mod n.
\]
RSA - Rivest, Shamir and Adleman

→ publicly described in 1977 by Rivest, Shamir, and Adleman at MIT

<table>
<thead>
<tr>
<th>public</th>
<th>private</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n = pq$</td>
<td>$d = e^{-1} \mod \phi(n)$</td>
</tr>
<tr>
<td>$e$ (public key)</td>
<td>$d$ (private key)</td>
</tr>
</tbody>
</table>

Encryption: $E(m) = m^e \mod n$

Decryption: $D(c) = c^d \mod n$

\[ m \xrightarrow{\text{encrypt}} m^e \mod n \xrightarrow{\text{decrypt}} (m^e \mod n)^d \mod n = m^{ed} \mod n \]

= $m \mod n$

RSA satisfies some other properties, e.g.

\[ E(m_1 \times m_2) = E(m_1) \times E(m_2) \]
Security of RSA

→ based on two mathematical problems:

Problem of factoring large numbers

the task of breaking down a composite number into smaller non-trivial divisors, which when multiplied together equal the original integer.

RSA problem

the task of taking $e^{th}$ roots modulo a composite $n$: recovering a value $m$ such that $c = m \times e \mod n$, where $(e, n)$ is an RSA public key and $c$ is an RSA ciphertext.
Security of RSA

→ based on two mathematical problems:

Problem of factoring large numbers

the task of breaking down a composite number into smaller non-trivial divisors, which when multiplied together equal the original integer.

RSA problem

the task of taking $e^{th}$ roots modulo a composite $n$: recovering a value $m$ such that $c = m \times e \mod n$, where $(e, n)$ is an RSA public key and $c$ is an RSA ciphertext.

→ As long as we are not able to solve these problems in an efficient way, the RSA scheme will be considered to be secure.
Symmetric vs. asymmetric encryption

Symmetric encryption
- efficient in practice,
- agents have to share a secret key
  → trusted third party, distribution key protocol

Asymmetric encryption
- not efficient in practice,
- agents do not have to share a secret
  → often used in establishment key protocols
- authentication of public keys (certificate)
Outline of the talk

1. Introduction

2. Cryptographic primitives
   - Symmetric encryption
   - Asymmetric encryption
   - Signature
   - Hash function

3. Cryptographic protocols
   - Toss of a coin by phone
   - A simple key distribution protocol
   - Needham-Schroeder public-key protocol
   - Credit Card payment
   - Zero-knowledge protocols

4. Conclusion
**Digital signature: How does it work?**

- similar to public key encryption
- everyone knows the key to verify the signature (public key)
- the key used to sign a message has to be private (private key)
Properties

- the signature has to authenticate the signer
- the signature “belongs to” one particular document
- the signed document can not be modified afterwards
Properties

- the signature has to authenticate the signer
- the signature “belongs to” one particular document
- the signed document cannot be modified afterwards

Applications

- contract signing protocols
- E-voting protocols (blind signature)
  → allows someone to sign without knowing the message he is signing.
Properties and applications

Properties

- the signature has to authenticate the signer
- the signature “belongs to“ one particular document
- the signed document can not be modified afterwards

Applications

- contract signing protocols
- E-voting protocols (blind signature)

→ allows someone to sign without knowing the message he is signing.

\[ v, r \text{ (blinding factor)} \xrightarrow{\text{voter}} bld(v, r) \xrightarrow{\text{adm}} sgn(bld(v, r), ska) \]

Then, the voter can obtain \( sgn(v, ska) \) by using \( r \).
Outline of the talk

1. Introduction

2. Cryptographic primitives
   - Symmetric encryption
   - Asymmetric encryption
   - Signature
   - Hash function

3. Cryptographic protocols
   - Toss of a coin by phone
   - A simple key distribution protocol
   - Needham-Schroeder public-key protocol
   - Credit Card payment
   - Zero-knowledge protocols

4. Conclusion
Hash function

It is a reproducible method of turning some kind of data into a (relatively) small number that may serve as a digital "fingerprint" of the data (again substitutions and permutations).

Examples: MD5, SHA-1
Properties

- deterministic function
- one-way function: there is no practical way to retrieve \( m \) from \( \text{hash}(m) \)
- collision resistant: difficult to find \( m_1 \) and \( m_2 \) such that \( m_1 \neq m_2 \) and \( \text{hash}(m_1) = \text{hash}(m_2) \)

Some applications

- to improve efficiency: we can sign \( \text{hash}(m) \) instead of \( m \)
- use to guarantee the integrity of a message
- checksum to detect errors
- toss of a coin by phone (later on)
Outline of the talk

1. Introduction

2. Cryptographic primitives
   - Symmetric encryption
   - Asymmetric encryption
   - Signature
   - Hash function

3. Cryptographic protocols
   - Toss of a coin by phone
   - A simple key distribution protocol
   - Needham-Schroeder public-key protocol
   - Credit Card payment
   - Zero-knowledge protocols

4. Conclusion
Toss of a coin by phone

Question?

How to choose who will do the washing-up this evening?
Bob chooses an odd or an even number, say $n$, and send $\text{hash}(n)$ to Alice.
Bob chooses an odd or an even number, say \( n \), and send \( \text{hash}(n) \) to Alice.

Alice tries to guess whether the number is odd or even. She sends her answer to Bob.
**Toss of a coin by phone**

Bob chooses an odd or an even number, say $n$, and send $\text{hash}(n)$ to Alice.

Alice tries to guess whether the number is odd or even. She sends her answer to Bob.

Bob answers whether Alice is true or not and send his number $n$ to Alice.
Toss of a coin by phone

Bob chooses an odd or an even number, say $n$, and send $\text{hash}(n)$ to Alice.

Alice tries to guess whether the number is odd or even. She sends her answer to Bob.

Bob answers whether Alice is true or not and send his number $n$ to Alice.

Alice verifies that Bob does not change his mind by computing $\text{hash}(n)$ to compare with the value she received at the beginning.
Outline of the talk

1. Introduction

2. Cryptographic primitives
   - Symmetric encryption
   - Asymmetric encryption
   - Signature
   - Hash function

3. Cryptographic protocols
   - Toss of a coin by phone
   - A simple key distribution protocol
   - Needham-Schroeder public-key protocol
   - Credit Card payment
   - Zero-knowledge protocols

4. Conclusion
A protocol based on commutative encryption (RSA)

\[\text{Alice:} \{\text{secret}\}_{\text{pub}(A)} \rightarrow \{\{\text{secret}\}_{\text{pub}(A)}\}_{\text{pub}(B)} \rightarrow \{\text{secret}\}_{\text{pub}(B)} \rightarrow \{\{\text{secret}\}_{\text{pub}(B)}\}_{\text{pub}(A)} = \{\{\text{secret}\}_{\text{pub}(A)}\}_{\text{pub}(B)}\]
A protocol based on commutative encryption (RSA)

This protocol does not work! (authentication problem)
A protocol based on commutative encryption (RSA)

\[ \text{pub}(A) \text{pub}(B) \]

This protocol does not work! (authentication problem)
Outline of the talk

1. Introduction

2. Cryptographic primitives
   - Symmetric encryption
   - Asymmetric encryption
   - Signature
   - Hash function

3. Cryptographic protocols
   - Toss of a coin by phone
   - A simple key distribution protocol
   - Needham-Schroeder public-key protocol
   - Credit Card payment
   - Zero-knowledge protocols

4. Conclusion
Needham-Schroeder’s Protocol (1978)

\[
\begin{align*}
A & \rightarrow B : \{A, N_a\}_{\text{pub}(B)} \\
B & \rightarrow A : \{N_a, N_b\}_{\text{pub}(A)} \\
A & \rightarrow B : \{N_b\}_{\text{pub}(B)}
\end{align*}
\]
Needham-Schroeder’s Protocol (1978)

\[ A \rightarrow B : \{ A, N_a \}_{\text{pub}(B)} \]
\[ B \rightarrow A : \{ N_a, N_b \}_{\text{pub}(A)} \]
\[ A \rightarrow B : \{ N_b \}_{\text{pub}(B)} \]
Needham-Schroeder’s Protocol (1978)

\[
A \rightarrow B : \quad \{A, N_a\}_{\text{pub}(B)} \\
B \rightarrow A : \quad \{N_a, N_b\}_{\text{pub}(A)} \\
\bullet \quad A \rightarrow B : \quad \{N_b\}_{\text{pub}(B)}
\]
Needham-Schroeder’s Protocol (1978)

\[
\begin{align*}
A & \rightarrow B : \quad \{A, N_a\}_{\text{pub}(B)} \\
B & \rightarrow A : \quad \{N_a, N_b\}_{\text{pub}(A)} \\
A & \rightarrow B : \quad \{N_b\}_{\text{pub}(B)}
\end{align*}
\]
Needham-Schroeder’s Protocol (1978)

\[\begin{align*}
A & \rightarrow B : \quad \{A, N_a\}_{\text{pub}(B)} \\
B & \rightarrow A : \quad \{N_a, N_b\}_{\text{pub}(A)} \\
A & \rightarrow B : \quad \{N_b\}_{\text{pub}(B)}
\end{align*}\]

Questions

- Is \(N_b\) secret between \(A\) and \(B\)?
- When \(B\) receives \(\{N_b\}_{\text{pub}(B)}\), does this message really come from \(A\)?
Needham-Schroeder’s Protocol (1978)

A → B : \{A, Na\}_{pub(B)}
B → A : \{Na, Nb\}_{pub(A)}
A → B : \{Nb\}_{pub(B)}

Questions

- Is $Nb$ secret between $A$ and $B$?
- When $B$ receives $\{Nb\}_{pub(B)}$, does this message really come from $A$?

Attack

An attack was found 17 years after its publication! [Lowe 96]
Example: Man in the middle attack

- Involving 2 sessions in parallel, an honest agent has to initiate a session with I.

\[
\begin{align*}
A \rightarrow B &: \{A, N_a\}_{\text{pub}(B)} \\
B \rightarrow A &: \{N_a, N_b\}_{\text{pub}(A)} \\
A \rightarrow B &: \{N_b\}_{\text{pub}(B)}
\end{align*}
\]
Example: Man in the middle attack

Agent $A$ \xrightarrow{{A, N_a}_{\text{pub}(I)}} \text{Intrus } I \xrightarrow{{A, N_a}_{\text{pub}(B)}} \text{Agent } B

A \rightarrow B : \{A, N_a\}_{\text{pub}(B)}
B \rightarrow A : \{N_a, N_b\}_{\text{pub}(A)}
A \rightarrow B : \{N_b\}_{\text{pub}(B)}
Example: Man in the middle attack

Agent $A$ → $B$ : $\{A, N_a\}_{pub(B)}$

Intrus $I$

Agent $B$ → $A$ : $\{N_a, N_b\}_{pub(A)}$

Agent $A$ → $B$ : $\{N_b\}_{pub(B)}$

Stéphanie Delaune (LSV)  Introduction to cryptographic protocols  42 / 66
Example: Man in the middle attack

\[
\begin{array}{ccc}
\text{Agent } A & \xrightarrow{\{A, N_a\}_{\text{pub}(I)}} & \text{Intrus } I \\
\xleftarrow{\{N_a, N_b\}_{\text{pub}(A)}} & & \xleftarrow{\{N_a, N_b\}_{\text{pub}(A)}} \\
\xrightarrow{\{N_b\}_{\text{pub}(I)}} & & \xrightarrow{\{N_b\}_{\text{pub}(B)}}
\end{array}
\]

\[
\begin{align*}
A & \rightarrow B : \{A, N_a\}_{\text{pub}(B)} \\
B & \rightarrow A : \{N_a, N_b\}_{\text{pub}(A)} \\
A & \rightarrow B : \{N_b\}_{\text{pub}(B)}
\end{align*}
\]
Example: Man in the middle attack

Agent A

\[
\{ A, N_a \}_{pub(I)} \rightarrow \{ N_a, N_b \}_{pub(A)} \rightarrow \{ N_b \}_{pub(I)}
\]

Intrus I

\[
\rightarrow \{ A, N_a \}_{pub(B)} \rightarrow \{ N_a, N_b \}_{pub(A)} \rightarrow \{ N_b \}_{pub(B)}
\]

Agent B

Attack

- the intruder knows \( N_b \),
- When B finishes his session (apparently with A), A has never talked with B.

\[
\begin{align*}
A \rightarrow B & : \{ A, N_a \}_{pub(B)} \\
B \rightarrow A & : \{ N_a, N_b \}_{pub(A)} \\
A \rightarrow B & : \{ N_b \}_{pub(B)}
\end{align*}
\]
Outline of the talk

1. Introduction

2. Cryptographic primitives
   - Symmetric encryption
   - Asymmetric encryption
   - Signature
   - Hash function

3. Cryptographic protocols
   - Toss of a coin by phone
   - A simple key distribution protocol
   - Needham-Schroeder public-key protocol
   - Credit Card payment
   - Zero-knowledge protocols

4. Conclusion
Example: credit card payment

- The client \( C_I \) puts his credit card \( C \) in the terminal \( T \).

- The merchant enters the amount \( M \) of the sale.

- The terminal authenticates the credit card.

- The client enters his PIN.
  - If \( M \geq 100 \€ \), then in 20% of cases,
    - The terminal contacts the bank \( B \).
    - The banks gives its authorisation.
the Bank $B$, the Client $C_l$, the Credit Card $C$ and the Terminal $T$
the Bank $B$, the Client $C_l$, the Credit Card $C$ and the Terminal $T$

Bank

- a private signature key – $\text{priv}(B)$
- a public key to verify a signature – $\text{pub}(B)$
- a secret key shared with the credit card – $K_{CB}$
the Bank $B$, the Client $Cl$, the Credit Card $C$ and the Terminal $T$

Bank
- a private signature key – $\text{priv}(B)$
- a public key to verify a signature – $\text{pub}(B)$
- a secret key shared with the credit card – $K_{CB}$

Credit Card
- some $Data$: name of the cardholder, expiry date …
- a signature of the $Data$ – $\{\text{hash}(Data)\}_\text{priv}(B)$
- a secret key shared with the bank – $K_{CB}$
More details

the Bank $B$, the Client $C$, the Credit Card $C$ and the Terminal $T$

**Bank**
- a **private** signature key – $\text{priv}(B)$
- a **public** key to verify a signature – $\text{pub}(B)$
- a **secret** key shared with the credit card – $K_{CB}$

**Credit Card**
- some *Data*: name of the cardholder, expiry date …
- a signature of the *Data* – $\{\text{hash}(\text{Data})\}_{\text{priv}(B)}$
- a **secret** key shared with the bank – $K_{CB}$

**Terminal**
- the **public** key of the bank – $\text{pub}(B)$
the terminal $T$ reads the credit card $C$:

1. $C \rightarrow T : \text{Data, } \{\text{hash(Data)}\}_{\text{priv}(B)}$
the terminal $T$ reads the credit card $C$:

1. $C \rightarrow T : \text{Data, } \{\text{hash(Data)}\}_{\text{priv(B)}}$

the terminal $T$ asks the code:

2. $T \rightarrow Cl : \text{code?}$
3. $Cl \rightarrow C : 1234$
4. $C \rightarrow T : \text{ok}$
Payment protocol

the terminal $T$ reads the credit card $C$:

1. $C \rightarrow T : \text{Data, } \{\text{hash(Data)}\}_{\text{priv}(B)}$

the terminal $T$ asks the code:

2. $T \rightarrow Cl : \text{code?}$
3. $Cl \rightarrow C : \text{1234}$
4. $C \rightarrow T : \text{ok}$

the terminal $T$ requests authorisation the bank $B$:

5. $T \rightarrow B : \text{auth?}$
6. $B \rightarrow T : \text{4528965874123}$
7. $T \rightarrow C : \text{4528965874123}$
8. $C \rightarrow T : \{\text{4528965874123}\}_{K_{CB}}$
9. $T \rightarrow B : \{\text{4528965874123}\}_{K_{CB}}$
10. $B \rightarrow T : \text{ok}$
the terminal $T$ asks the code:

2. $T \rightarrow Cl : \text{code?}$
3. $Cl \rightarrow C : 1234$
4. $C \rightarrow T : \text{ok}$
the terminal $T$ asks the code:

2. $T \rightarrow Cl : \text{code?}$
3. $Cl \rightarrow C : 1234$
4. $C \rightarrow T : \text{ok}$

Discussion

- the secret code is revealed to the verifier
- the secret code is revealed to any eavesdropper
- the verifier of the proof is the credit card itself
  $\rightarrow$ Yes Card – (Serge Humpich case)
the terminal $T$ requests authorisation to the bank $B$:

5. $T \rightarrow B : \text{auth?}$
6. $B \rightarrow T : 4528965874123$
7. $T \rightarrow C : 4528965874123$
8. $C \rightarrow T : \{4528965874123\}K_{CB}$
9. $T \rightarrow B : \{4528965874123\}K_{CB}$
10. $B \rightarrow T : \text{ok}$
Requesting authorisation to the bank

the terminal $T$ requests authorisation to the bank $B$:

5. $T \rightarrow B : \text{auth}$
6. $B \rightarrow T : 4528965874123$
7. $T \rightarrow C : 4528965874123$
8. $C \rightarrow T : \{4528965874123\} K_{CB}$
9. $T \rightarrow B : \{4528965874123\} K_{CB}$
10. $B \rightarrow T : \text{ok}$

Discussion

- the secret code is already known by the verifier
- challenge mecanism: the prover $C$ proves to the verifier $B$ that he knows the secret key $K_{CB}$
- an eavesdropper does not learn the secret $K_{CB}$ but he learns something about it
Outline of the talk

1. Introduction

2. Cryptographic primitives
   - Symmetric encryption
   - Asymmetric encryption
   - Signature
   - Hash function

3. Cryptographic protocols
   - Toss of a coin by phone
   - A simple key distribution protocol
   - Needham-Schroeder public-key protocol
   - Credit Card payment
   - Zero-knowledge protocols

4. Conclusion
Zero-knowledge proofs are proofs that are both convincing and yet yield nothing beyond the validity of the assertion being proved.

→ introduced 20 years ago by Goldwasser, Micali and Rackoff [1985]
Zero-knowledge proofs are proofs that are both convincing and yet yield nothing beyond the validity of the assertion being proved.

→ introduced 20 years ago by Goldwasser, Micali and Rackoff [1985]

- **Completeness**: if the statement is true, the honest verifier will be convinced of this fact by an honest prover.

- **Soundness**: if the statement is false, no cheating prover can convince the honest verifier that it is true.
Zero-knowledge proofs are proofs that are both convincing and yet yield nothing beyond the validity of the assertion being proved.

→ introduced 20 years ago by Goldwasser, Micali and Rackoff [1985]

- **Completeness**: if the statement is true, the honest verifier will be convinced of this fact by an honest prover.

- **Soundness**: if the statement is false, no cheating prover can convince the honest verifier that it is true.

- **Zero-knowledge**: If the statement is true, no cheating verifier learns anything other than this fact.
Zero-knowledge proofs are proofs that are both convincing and yet yield nothing beyond the validity of the assertion being proved.

→ introduced 20 years ago by Goldwasser, Micali and Rackoff [1985]

- **Completeness**: if the statement is true, the honest verifier will be convinced of this fact by an honest prover.
- **Soundness**: if the statement is false, no cheating prover can convince the honest verifier that it is true.
- **Zero-knowledge**: If the statement is true, no cheating verifier learns anything other than this fact.

The definitions given above seem to be contradictory.

→ Does zero-knowledge proofs really exist?
Applications

Authentication properties

- Credit card payment
  → to prove that you know the secret code without revealing it
- prove your identity
- prove that you belongs to a group without revealing who you are
  → to ensure privacy

Other properties

- to enforce honest behavior
  → e.g. mix net in electronic voting protocols,
- ....
Example: Where is Charlie?

Goal:

1. find the reporter Charlie in a big picture,
2. convince the verifier (me) that you have the solution without revealing it (neither to me, nor to the others).
Example: Where is Charlie?
Example: Where is Charlie?

How can you prove that you know where is Charlie without saying nothing about where he is?

Solutions:
Example: Where is Charlie?

How can you prove that you know where is Charlie without saying nothing about where he is?

Solutions:

1. get a copy of the picture, cut out Charlie and show it to me.
2. put a big mask with a window having the shape of Charlie and show me Charlie through the window.
Example: The strange cave of Ali Baba

- a cave shaped like a circle, with entrance on one side and the magic door blocking the opposite side
- the door can be opened by saying some magic words “....”.

Goal:

Ali Baba wants to convince me that he knows the secret without revealing it.

How can Ali Baba proceed?
Example: The strange cave of Ali Baba

Ali Baba wants to convince me that he knows the magic words.

Ali Baba hides inside the cave

I ask him to exit on the right side or on the left side

→ I choose

Ali Baba exits from the side I just asked.
Example: The strange cave of Ali Baba

Ali Baba wants to convince me that he knows the magic words.

Ali Baba hides inside the cave

I ask him to exit on the right side or on the left side
→ I choose

Ali Baba exits from the side I just asked.

... and we repeat this procedure several times.
Example: The strange cave of Ali Baba

I can be convinced that Ali Baba knows the magic words.

Why?

- If Ali Baba does not know the magic word, then he can only return by the same path. Since, I randomly choose the path, he has 50% chance of guessing correctly.

- By repeating this trick many times, say 20 times, his chance of successfully anticipating all my requests becomes very small.
Example: The strange cave of Ali Baba

I can be convinced that Ali Baba knows the magic words.

Why?

- If Ali Baba does not know the magic word, then he can only return by the same path. Since, I randomly choose the path, he has 50% chance of guessing correctly.

- By repeating this trick many times, say 20 times, his chance of successfully anticipating all my requests becomes very small.

Moreover,

- I learn nothing about the magic word beyond the fact this word allows Ali Baba to open the magic door, and

- I am not able to prove to someone else that I know the magic words.
Example: The strange cave of Ali Baba

The End
– of Ali Baba story –

To know the magic word:

*How to explain Zero-Knowledge Protocols to Your Children.*
Jean-Jacques Quisquater and Louis Guillou.
### Definition (3-coloring)

A **3-coloring** of a graph is an assignment of colors in \{\textbullet, \textcolor{red}{\textbullet}, \textcolor{green}{\textbullet}\} to vertices such that no pair of adjacent vertices are assigned to the same color.
Definition (3-coloring)

A 3-coloring of a graph is an assignment of colors in \{\color{blue}{\bullet}, \color{red}{\bullet}, \color{green}{\bullet}\} to vertices such that no pair of adjacent vertices are assigned to the same color.

Example
A 3-coloring of a graph is an assignment of colors in \{\textbullet, \textcolor{red}{\textbullet}, \textcolor{green}{\textbullet}\} to vertices such that no pair of adjacent vertices are assigned to the same color.

**Example**
Definition (3-coloring)

A 3-coloring of a graph is an assignment of colors in \{\textcolor{red}{\bullet}, \textcolor{red}{\bullet}, \textcolor{red}{\bullet}\} to vertices such that no pair of adjacent vertices are assigned to the same color.

Example

3-coloring problem

Given graph \( G \), the problem of deciding if the graph \( G \) is 3-colorable is a very hard problem. It is also very hard to find a 3-coloring of a large graph.
Protocol based on the 3-coloring problem
Protocol based on the 3-coloring problem
Protocol based on the 3-coloring problem
Protocol based on the 3-coloring problem

\[ \{ \bullet \}_k, \{ \circ \}_k, \{ \circ \}_k, \{ \bullet \}_k, \{ \circ \}_k, \{ \bullet \}_k \]
Protocol based on the 3-coloring problem

Choose an edge $(1, 4)$
Protocol based on the 3-coloring problem

Choose an edge (1, 4)

Send keys $k_1$ and $k_4$
Protocol based on the 3-coloring problem

choose an edge

(1, 4)

send keys

decrypt \{○\}_k with k_1

decrypt \{•\}_k with k_4

accept since • ≠ ○

... repeat the procedure several times
Discussion on the protocol

- **Completeness**: if the statement is true, the honest verifier will be convinced of this fact by an honest prover.

  → if Ali Baba knows the 3-coloring of the graph, then the verifier will accept his proof.
Discussion on the protocol

- **Completeness**: if the statement is true, the honest verifier will be convinced of this fact by an honest prover.
  
  → if Ali Baba knows the 3-coloring of the graph, then the verifier will accept his proof.

- **Soundness**: if the statement is false, no cheating prover can convince the honest verifier that it is true.
  
  → if Ali Baba does not know a 3-coloring of the graph, then Bob rejects with probability \( \frac{1}{\# edges} \).
Discussion on the protocol

- **Completeness**: if the statement is true, the honest verifier will be convinced of this fact by an honest prover.
  
  → if Ali Baba knows the 3-coloring of the graph, then the verifier will accept his proof.

- **Soundness**: if the statement is false, no cheating prover can convince the honest verifier that it is true.
  
  → if Ali Baba does not know a 3-coloring of the graph, then Bob rejects with probability $\frac{1}{\#edges}$.

- **Zero-knowledge**: If the statement is true, no cheating verifier learns anything other than this fact.
  
  → Bob just sees two random colors. Hence, he learns nothing about the 3-coloring of the graph.
Composability

This allows us to use the same protocol several times.
Composability

This allows us to use the same protocol several times.

**Sequential composition**
Each invocation follows the termination of the previous one.

→ Generally, sequential composition is **safe**. Note that **otherwise**, the applicability of the protocol is **highly limited**.
Composability

This allows us to use the same protocol several times.

**Sequential composition**
Each invocation follows the termination of the previous one.

→ Generally, sequential composition is safe. Note that otherwise, the applicability of the protocol is highly limited.

**Parallel composition**
Many instances of the protocol are invoked at the same time and proceed at the same pace (synchronous model of communication)

→ Generally, parallel composition is not safe.
Composability

This allows us to use the same protocol several times.

**Sequential composition**
Each invocation follows the termination of the previous one.

→ Generally, sequential composition is **safe**. Note that **otherwise**, the applicability of the protocol is **highly limited**.

**Parallel composition**
Many instances of the protocol are invoked at the same time and proceed at the same pace (synchronous model of communication)

→ Generally, parallel composition is **not safe**.

**Concurrent composition**
This generalizes both sequential and parallel composition. Many instances of the protocol are invoked at arbitrary times and proceed at arbitrary pace.
Parallel composition – Man in the middle attack

The **intruder** wants to convince **Bob** that he knows the **secret**.

How can he do this?

I know the **secret**
Parallel composition – Man in the middle attack

The **intruder** wants to convince **Bob** that he knows the **secret**.

I know the **secret**

How can he do this?

question
Parallel composition – Man in the middle attack

The **intruder** wants to convince **Bob** that he knows the **secret**.

I know the **secret**

How can he do this?

**question** → **question** → **answer**
Parallel composition – Man in the middle attack

The intruder wants to convince Bob that he knows the secret.

How can he do this?

I know the secret

question

answer

question

answer
Parallel composition – Man in the middle attack

The intruder wants to convince Bob that he knows the secret.

I know the secret

How can he do this?

question

question

answer

answer

→ This kind of attack often succeeds on Zero-knowledge protocols.
Outline of the talk

1. Introduction

2. Cryptographic primitives
   - Symmetric encryption
   - Asymmetric encryption
   - Signature
   - Hash function

3. Cryptographic protocols
   - Toss of a coin by phone
   - A simple key distribution protocol
   - Needham-Schroeder public-key protocol
   - Credit Card payment
   - Zero-knowledge protocols

4. Conclusion
Conclusion

Cryptographic protocols

- numerous, various security goals
- can be attacked even if the primitives are secure
  \[\rightarrow \text{http://www.lsv.ens-cachan.fr/spore/}\]

How to verify them?

- modelling the protocol, the security properties
- manually / automatically
  \[\rightarrow \text{the problem is undecidable in general (some tools exist)}\]

It remains a lot to do

- modelling security properties is a difficult task
- does a suitable E-voting protocol exist?
- take into account the algebraic properties of the primitives
- analyse the source code of the protocol instead of its specification
1. Wikipedia web site
2. Security Protocols Open Repository (SPORE)
   http://www.lsv.ens-cachan.fr/spore/
3. *How to explain Zero-Knowledge Protocols to Your Children.*
   Jean-Jacques Quisquater and Louis Guillou.