

# Introduction to cryptographic protocols

Stéphanie Delaune

September 17, 2008



## 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?
- **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



Belgique - Election 2004 - <http://www.poueva.be/> - (C) Kanar

## 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, **encryption** and **signature** functions.

## Cryptographic primitives

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

### Symmetric encryption



→ **Examples:** Caesar encryption, DES, AES, ...

# Cryptographic primitives

## Cryptographic primitives

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

### Asymmetric encryption

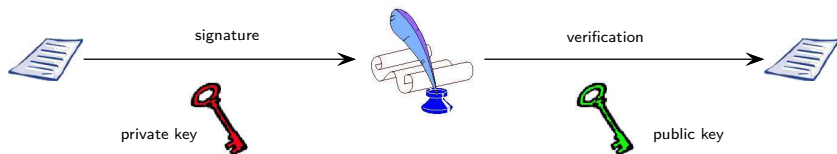


# Cryptographic primitives

## Cryptographic primitives

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

### Signature



# Security of cryptographic protocols

How cryptographic protocols can be attacked?

Breaking encryption



cryptanalysis

Logical attack



replay attacks

man-in-the-middle attacks

# Replay attacks



transfer 100 euros into  
the merchant's account

---



# Replay attacks



transfer 100 euros into  
the merchant's account

→



transfer 100 euros into  
the merchant's account

→



# Replay attacks



transfer 100 euros into  
the merchant's account



transfer 100 euros into  
the merchant's account



transfer 100 euros into  
the account's merchant



⋮

transfer 100 euros into  
the account's merchant



# Replay attacks



transfer 100 euros into  
the merchant's account



transfer 100 euros into  
the merchant's account



transfer 100 euros into  
the account's merchant

⋮

transfer 100 euros into  
the account's merchant



**Example:** attack on the decoders (TV)

→ block the message that cancels the subscription

# Logical attacks - How to detect them?

## Symbolic approach

- **messages** are represented by **terms** rather than bit-strings
  - ↔  $\{m\}_k$  encryption of the message  $m$  with key  $k$ ,
  - ↔  $\langle m_1, m_2 \rangle$  pairing of messages  $m_1$  and  $m_2$ , ...
- **attacker** controls the network and can perform **specific actions**

# Logical attacks - How to detect them?

## Symbolic approach

- **messages** are represented by **terms** rather than bit-strings
  - ↦  $\{m\}_k$  encryption of the message  $m$  with key  $k$ ,
  - ↦  $\langle m_1, m_2 \rangle$  pairing of messages  $m_1$  and  $m_2$ , ...
- **attacker** controls the network and can perform **specific actions**

## Relevance of the approach

- **numerous** attacks have already been obtained,
- **soundness results** w.r.t. to the **computational model** already exist, e.g. [Abadi & Rogaway'01]
- allows us to perform **automatic** verification



## Credit Card Payment Protocol



# Example: credit card payment



- The client  $C$  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  $M \geq 100 \text{ €}$ , then in 20% of cases,
  - The terminal contacts the bank  $B$ .
  - The banks gives its authorisation.



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

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 –  $K_{CB}$

# More details

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** –  $\{\text{hash}(\text{Data})\}_{\text{priv}(B)}$
- a **secret** key shared with the bank –  $K_{CB}$

# More details

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** –  $\{\text{hash}(\text{Data})\}_{\text{priv}(B)}$
- a **secret** key shared with the bank –  $K_{CB}$

## Terminal

- the **public** key of the bank –  $\text{pub}(B)$

# Payment protocol

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

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

# Payment protocol

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

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

the terminal  $T$  asks the code:

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

# Payment protocol

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

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

the terminal  $T$  asks the code:

2.  $T \rightarrow CI : \text{code?}$

3.  $CI \rightarrow C : 1234$

4.  $C \rightarrow T : \text{ok}$

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

5.  $T \rightarrow B : \text{auth?}$

6.  $B \rightarrow T : 4528965874123$

7.  $T \rightarrow C : 4528965874123$

8.  $C \rightarrow T : \{4528965874123\}_{K_{CB}}$

9.  $T \rightarrow B : \{4528965874123\}_{K_{CB}}$

10.  $B \rightarrow T : \text{ok}$

# Faible sur la carte bleue

Initialement la sécurité été assurée par :

- cartes difficilement répliquables,
- secret des clefs et du protocole.



# Faible sur la carte bleue

Initialement la sécurité été assurée par :

- cartes difficilement répliquables,
- secret des clefs et du protocole.



Mais il y a des failles !

- faille **cryptographique** : les clefs de 320 bits ne sont plus sûres,
- faille **logique** : pas de lien entre le code secret à 4 chiffres et l'authentification,
- faille **matériel** : répliquabilité des cartes.



→ “YesCard” fabriquées par Serge Humpich (1997).

# La « YesCard » : Comment ça marche ?

## Faible logique

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

2.  $T \rightarrow Cl$  : *code?*

3.  $Cl \rightarrow C$  : 1234

4.  $C \rightarrow T$  : *ok*

# La « YesCard » : Comment ça marche ?

## Faible logique

1.  $C \rightarrow T$  :  $\text{Data}, \{\text{hash}(\text{Data})\}_{\text{priv}(B)}$
2.  $T \rightarrow CI$  : *code?*
3.  $CI \rightarrow C'$  : **2345**
4.  $C' \rightarrow T$  : *ok*

# La « YesCard » : Comment ça marche ?

## Faible logique

1.  $C \rightarrow T$  :  $\text{Data}, \{\text{hash}(\text{Data})\}_{\text{priv}(B)}$
2.  $T \rightarrow CI$  : *code?*
3.  $CI \rightarrow C'$  : **2345**
4.  $C' \rightarrow T$  : *ok*

**Remarque** : il y a toujours quelqu'un à débiter.

→ ajout d'un faux chiffrage sur une fausse carte (Serge Humpich).

# La « YesCard » : Comment ça marche ?

## Faible logique

1.  $C \rightarrow T$  :  $\text{Data}, \{\text{hash}(\text{Data})\}_{\text{priv}(B)}$
2.  $T \rightarrow CI$  : *code?*
3.  $CI \rightarrow C'$  : 2345
4.  $C' \rightarrow T$  : *ok*

**Remarque** : il y a toujours quelqu'un à débiter.

→ ajout d'un faux chiffrement sur une fausse carte (Serge Humpich).

1.  $C' \rightarrow T$  :  $\text{XXX}, \{\text{hash}(\text{XXX})\}_{\text{priv}(B)}$
2.  $T \rightarrow CI$  : *code?*
3.  $CI \rightarrow C'$  : 0000
4.  $C' \rightarrow T$  : *ok*

# Needham-Schroeder (public-key) Protocol

# Needham-Schroeder's Protocol (1978)



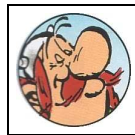
- $A \rightarrow B : \{A, N_a\}_{\text{pub}(B)}$   
 $B \rightarrow A : \{N_a, N_b\}_{\text{pub}(A)}$   
 $A \rightarrow B : \{N_b\}_{\text{pub}(B)}$



# Needham-Schroeder's Protocol (1978)



- $A \rightarrow B : \{A, N_a\}_{\text{pub}(B)}$   
 $B \rightarrow A : \{N_a, N_b\}_{\text{pub}(A)}$   
 $A \rightarrow B : \{N_b\}_{\text{pub}(B)}$



# Needham-Schroeder's Protocol (1978)



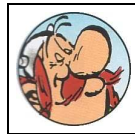
$A \rightarrow B : \{A, N_a\}_{\text{pub}(B)}$   
 $B \rightarrow A : \{N_a, N_b\}_{\text{pub}(A)}$   
•  $A \rightarrow B : \{N_b\}_{\text{pub}(B)}$



# Needham-Schroeder's Protocol (1978)



$A \rightarrow B : \{A, N_a\}_{\text{pub}(B)}$   
 $B \rightarrow A : \{N_a, N_b\}_{\text{pub}(A)}$   
 $A \rightarrow B : \{N_b\}_{\text{pub}(B)}$



# Needham-Schroeder's Protocol (1978)


$$\begin{aligned} A &\rightarrow B : \{A, N_a\}_{\text{pub}(B)} \\ B &\rightarrow A : \{N_a, N_b\}_{\text{pub}(A)} \\ A &\rightarrow B : \{N_b\}_{\text{pub}(B)} \end{aligned}$$


## Questions

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

# Needham-Schroeder's Protocol (1978)



$A \rightarrow B : \{A, N_a\}_{\text{pub}(B)}$   
 $B \rightarrow A : \{N_a, N_b\}_{\text{pub}(A)}$   
 $A \rightarrow B : \{N_b\}_{\text{pub}(B)}$



## Questions

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

## Attack

An attack was found 17 years after its publication! [Lowe 96]

# Example: Man in the middle attack



Agent A



Intruder I



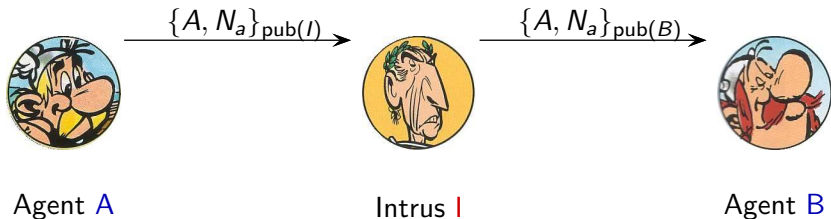
Agent B

## Attack

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

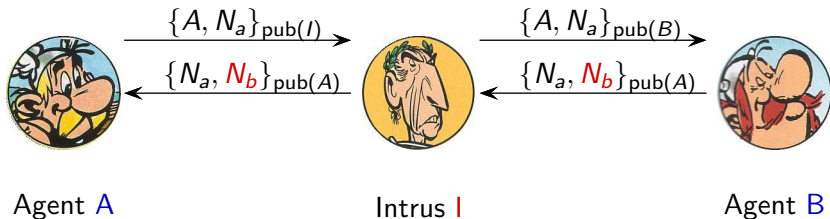
$$\begin{aligned} A \rightarrow B & : \{A, N_a\}_{\text{pub}(B)} \\ B \rightarrow A & : \{N_a, N_b\}_{\text{pub}(A)} \\ A \rightarrow B & : \{N_b\}_{\text{pub}(B)} \end{aligned}$$

# Example: Man in the middle attack



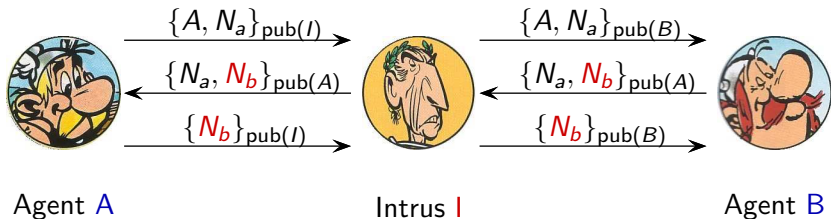
$A \rightarrow B : \{A, N_a\}_{\text{pub}(B)}$   
 $B \rightarrow A : \{N_a, N_b\}_{\text{pub}(A)}$   
 $A \rightarrow B : \{N_b\}_{\text{pub}(B)}$

# Example: Man in the middle attack



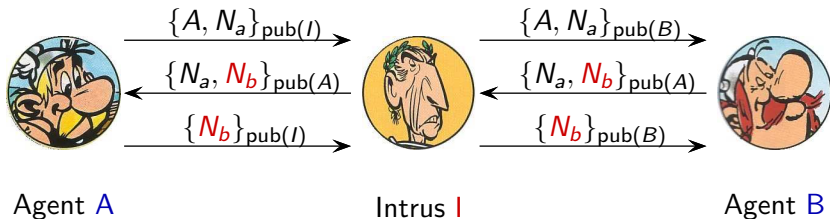
$A \rightarrow B : \{A, N_a\}_{\text{pub}(B)}$   
 $B \rightarrow A : \{N_a, N_b\}_{\text{pub}(A)}$   
 $A \rightarrow B : \{N_b\}_{\text{pub}(B)}$

# Example: Man in the middle attack



$A \rightarrow B : \{A, N_a\}_{\text{pub}(B)}$   
 $B \rightarrow A : \{N_a, N_b\}_{\text{pub}(A)}$   
 $A \rightarrow B : \{N_b\}_{\text{pub}(B)}$

# Example: Man in the middle attack



## Attack

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

$A \rightarrow B : \{A, N_a\}_{\text{pub}(B)}$   
 $B \rightarrow A : \{N_a, N_b\}_{\text{pub}(A)}$   
 $A \rightarrow B : \{N_b\}_{\text{pub}(B)}$

How to verify security protocols?

## Formal model (Dolev-Yao model)

- passive attacker: deduction / indistinguishability
- active attacker: bounded number of sessions  
unbounded number of sessions (Bruno Blanchet)

↕ link between the two models (passive case)

## Computational model

→ Bruno Blanchet

## Model and implementation (written in ML)

→ Cédric Fournet