Introduction to cryptographic protocols

Bruno Blanchet

INRIA, École Normale Supérieure, CNRS
blanchet@di.ens.fr

September 2011

(Partly based on slides by Stéphanie Delaune)

Cryptographic protocols

Paiement Internet

Cryptographic protocols

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

---

Cryptographic protocols

Paiement Internet

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.poureva.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

Cryptographic primitives

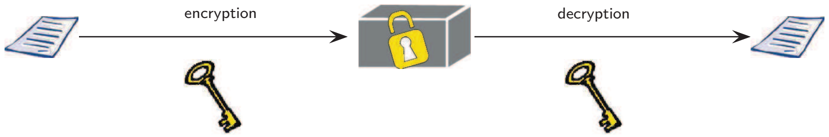
Algorithms that are frequently used to build computer security systems. These routines include, but are not limited to, encryption and signature functions.

Cryptographic primitives

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



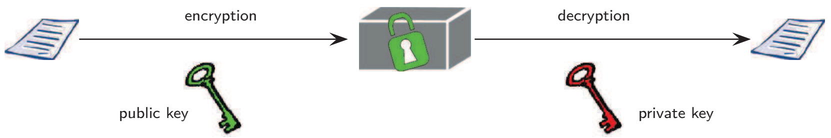
⟶ Examples: Caesar encryption, DES, AES, . . .

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

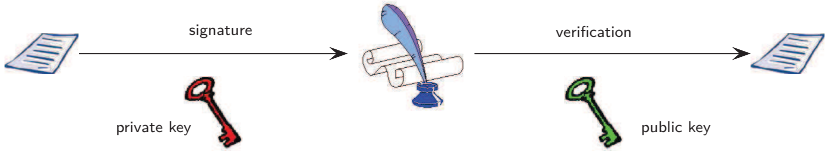


Cryptographic primitives

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



Why verify security protocols ?

The verification of security protocols has been and is still a very active research area.

- Their design is error prone.
- Security errors are not detected by testing: they appear only in the presence of an adversary.
- Errors can have serious consequences.

Models of protocols

Active attacker:

- the attacker can intercept all messages sent on the network
- he can compute messages
- he can send messages on the network

Models of protocols: the formal model

The formal model or “Dolev-Yao model” is due to Needham and Schroeder [1978] and Dolev and Yao [1983].

- The cryptographic primitives are blackboxes.
- The messages are terms on these primitives.
  $\hookrightarrow$ $\{m\}_k$ encryption of the message m with key k,
  $\hookrightarrow$ (m_1, m_2) pairing of messages m_1 and m_2, . . .
- The attacker is restricted to compute only using these primitives.
  $\Rightarrow$ perfect cryptography assumption

One can add equations between primitives, but in any case, one makes the hypothesis that the only equalities are those given by these equations.

This model makes automatic proofs relatively easy (AVISPA, ProVerif, . . .).

Models of protocols: the computational model

The computational model has been developed at the beginning of the 1980’s by Goldwasser, Micali, Rivest, Yao, and others.

- The messages are bitstrings.
- The cryptographic primitives are functions on bitstrings.
- The attacker is any probabilistic (polynomial-time) Turing machine.

This model is much more realistic than the formal model, but until recently proofs were only manual.

Models of protocols: side channels

The computational model is still just a model, which does not exactly match reality.

In particular, it ignores side channels:

- timing
- power consumption
- noise
- physical attacks against smart cards

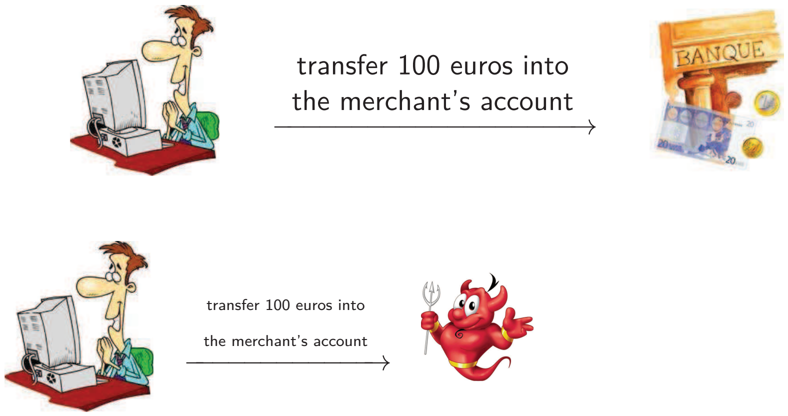
which can give additional information.

In this course, we will mostly ignore side channels.

Formal model: example of attacks, replay attacks

transfer 100 euros into the merchant’s account

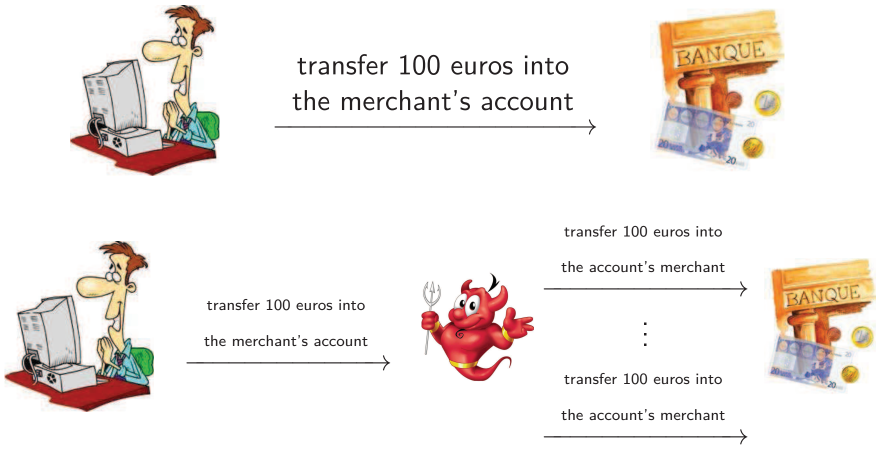
Formal model: example of attacks, replay attacks



transfer 100 euros into
the merchant's account

transfer 100 euros into
the merchant's account

Formal model: example of attacks, replay attacks



transfer 100 euros into
the merchant's account

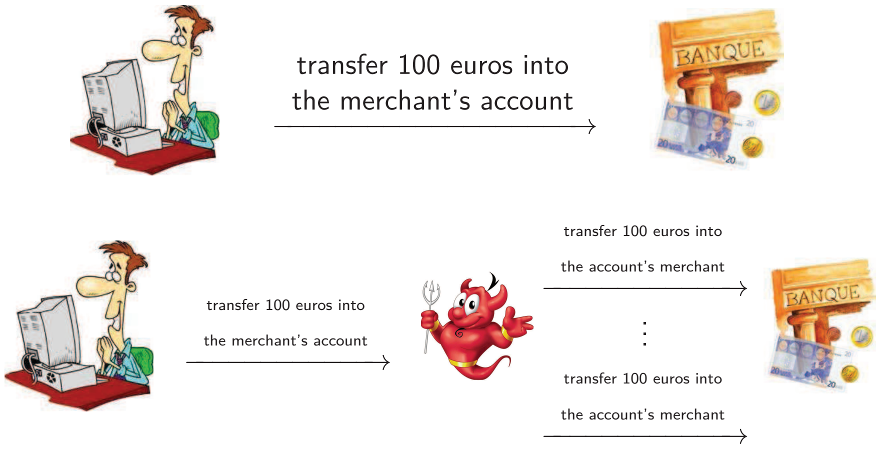
transfer 100 euros into
the merchant's account

transfer 100 euros into
the account's merchant

⋮

transfer 100 euros into
the account's merchant

Formal model: example of attacks, 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

Verifying protocols in the formal model

- Compute the set of all terms that the attacker can obtain.
- This set is infinite:
  - The attacker can generate messages of unbounded size.
  - The number of sessions of the protocol is unbounded.

Complexity

- Bounded messages and number of sessions
  - ⇒ finite state
  - Model checking: FDR [Lowe, TACAS'96]
- Bounded number of sessions but unbounded messages
  - ⇒ insecurity is typically NP-complete
  - Constraint solving: Cl-AtSe, integrated in AVISPA
    Extensions of model checking: OFMC, integrated in AVISPA
- Unbounded messages and number of sessions
  - ⇒ the problem is undecidable

Solutions to undecidability

- Rely on user interaction
  - Interactive theorem proving, Isabelle [Paulson, JCS'98]
- Use approximations
  - Abstract interpretation [Monniaux, SCP'03], TA4SP integrated in AVISPA
  - Typing [Abadi, JACM'99], [Gordon, Jeffrey, CSFW'02]
    (Sometimes also relies on type annotations by the user.)
- Allow non-termination

ProVerif uses approximations and allows non-termination.

Relevance of the formal model

- Numerous attacks have already been obtained.
- An attack in the formal model immediately implies an in the computational model (and a practical attack).
  - A proof in the formal model does not always imply a proof in the computational model (see next).
- Allows us to perform automatic verification.

Proofs in the computational model

- Manual proofs by cryptographers:
  - proofs by sequences of games [Shoup, Bellare&Rogaway]
- Automation:
  - CryptoVerif
  - CertiCrypt, framework within Coq
  - Typing

Link between the two models

- Computational soundness theorems:

  Proof in the formal model ⇒ proof in the computational model

  modulo additional assumptions.

  Approach pioneered by Abadi&Rogaway [2000]; many works since then.

Link between the two models: application

- Indirect approach to automating computational proofs:

1. Automatic formal protocol verifier

↓

proof in the formal model

2. Computational soundness

⟶

proof in the computational model

Credit Card Payment Protocol

Example: credit card payment

- The client Cl 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.

More details

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

More details

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

Bank

- a private signature key – $\mathrm{priv}(B)$
- a public key to verify a signature – $\mathrm{pub}(B)$
- 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 – $\mathrm{priv}(B)$
- a public key to verify a signature – $\mathrm{pub}(B)$
- 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 – $\{\mathit{hash}(\mathit{Data})\}_{\mathrm{priv}(B)}$
- 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 – $\mathrm{priv}(B)$
- a public key to verify a signature – $\mathrm{pub}(B)$
- 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 – $\{\mathit{hash}(\mathit{Data})\}_{\mathrm{priv}(B)}$
- a secret key shared with the bank – K_{CB}

Terminal

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

Payment protocol

the terminal T reads the credit card C:

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

Payment protocol

the terminal T reads the credit card C:

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

the terminal T asks the code:

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

Payment protocol

the terminal T reads the credit card C:

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

the terminal T asks the code:

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

the terminal T requests authorisation the bank B:

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

Attack against credit cards

Initially, security was guaranteed by:

- cards hard to replicate,
- secrecy of keys and protocol.

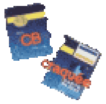
Attack against credit cards

Initially, security was guaranteed by:

- cards hard to replicate,
- secrecy of keys and protocol.

However, there are attacks!

- cryptographic attack: 320-bit keys are no longer secure,
- logical attack: no link between the 4-digit PIN code and the authentication,
- hardware attack: replication of cards.



→ "YesCard" made by Serge Humpich (1997).

The « YesCard »: how does it work?

Logical attack

1.$C \to T$: Data, $\{\text{hash}(\text{Data})\}_{\text{priv}(B)}$
2.$T \to CI$: $PIN?$
3.$CI \to C$: 1234
4.$C \to T$: ok

The « YesCard »: how does it work?

Logical attack

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

The « YesCard »: how does it work?

Logical attack

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

Remark: there is always somebody to debit.
→ add a fake ciphertext on a fake card (Serge Humpich).

The « YesCard »: how does it work?

Logical attack

$$\begin{array}{llll} 1. C & \rightarrow T & : \text{Data}, \{\text{hash}(\text{Data})\}_{\text{priv}(B)} \\ 2. T & \rightarrow CI & : PIN? \\ 3. CI & \rightarrow C' & : 2345 \\ 4. C' & \rightarrow T & : ok \end{array}$$

Remark: there is always somebody to debit.
→ add a fake ciphertext on a fake card (Serge Humpich).

$$\begin{array}{llll} 1. C' & \rightarrow T & : \text{XXX}, \{\text{hash}(\text{XXX})\}_{\text{priv}(B)} \\ 2. T & \rightarrow CI & : PIN? \\ 3. CI & \rightarrow C' & : 0000 \\ 4. C' & \rightarrow T & : ok \end{array}$$

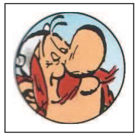
Needham-Schroeder (public-key) Protocol

Needham-Schroeder's Protocol (1978)

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

Needham-Schroeder's Protocol (1978)

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

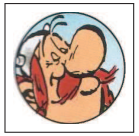


Needham-Schroeder's Protocol (1978)

$$\begin{array}{rlll} & A & \rightarrow & B: & \{A, N_a\}_{\mathrm{pub}(B)} \\ & B & \rightarrow & A: & \{N_a, N_b\}_{\mathrm{pub}(A)} \\ \bullet & A & \rightarrow & B: & \{N_b\}_{\mathrm{pub}(B)} \end{array}$$

Needham-Schroeder's Protocol (1978)

$$\begin{array}{llll} A & \rightarrow & B: & \{A, N_a\}_{\mathrm{pub}(B)} \\ B & \rightarrow & A: & \{N_a, N_b\}_{\mathrm{pub}(A)} \\ A & \rightarrow & B: & \{N_b\}_{\mathrm{pub}(B)} \end{array}$$



Needham-Schroeder's Protocol (1978)

$$\begin{array}{llll} A & \rightarrow & B: & \{A, N_a\}_{\mathrm{pub}(B)} \\ B & \rightarrow & A: & \{N_a, N_b\}_{\mathrm{pub}(A)} \\ A & \rightarrow & B: & \{N_b\}_{\mathrm{pub}(B)} \end{array}$$

Questions

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

Needham-Schroeder's Protocol (1978)

$$\begin{array}{llll} A & \rightarrow & B: & \{A, N_a\}_{\mathrm{pub}(B)} \\ B & \rightarrow & A: & \{N_a, N_b\}_{\mathrm{pub}(A)} \\ A & \rightarrow & B: & \{N_b\}_{\mathrm{pub}(B)} \end{array}$$

Questions

- Is N_b secret between A and B ?
- When B receives $\{N_b\}_{\mathrm{pub}(B)}$, 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

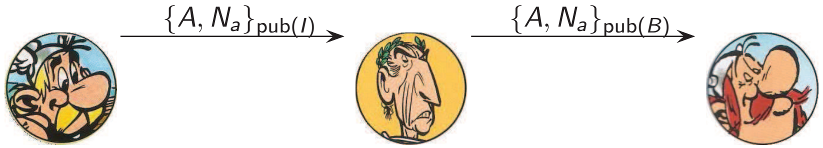
Agent B

Attack

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

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

Example: Man in the middle attack



$\{A, N_a\}_{\text{pub}(I)}$ →

$\{A, N_a\}_{\text{pub}(B)}$ →

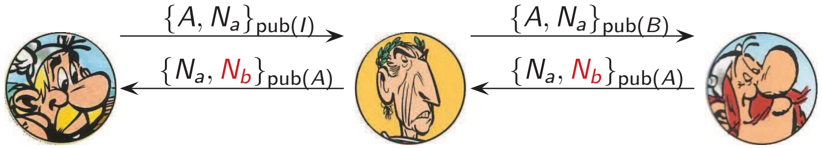
Agent A

Intruder I

Agent B

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

Example: Man in the middle attack



$\{A, N_a\}_{\text{pub}(I)}$ →

← $\{N_a, N_b\}_{\text{pub}(A)}$

$\{A, N_a\}_{\text{pub}(B)}$ →

← $\{N_a, N_b\}_{\text{pub}(A)}$

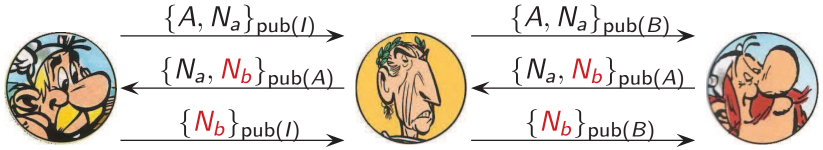
Agent A

Intruder I

Agent B

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

Example: Man in the middle attack



$\{A, N_a\}_{\text{pub}(I)}$ →

← $\{N_a, N_b\}_{\text{pub}(A)}$

$\{N_b\}_{\text{pub}(I)}$ →

$\{A, N_a\}_{\text{pub}(B)}$ →

← $\{N_a, N_b\}_{\text{pub}(A)}$

$\{N_b\}_{\text{pub}(B)}$ →

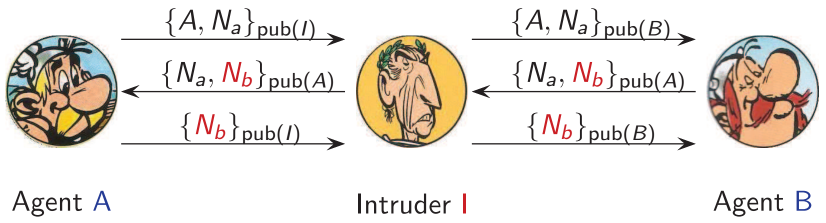
Agent A

Intruder I

Agent B

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

Example: Man in the middle attack



$\{A, N_a\}_{\text{pub}(I)} \rightarrow$

$\leftarrow \{N_a, N_b\}_{\text{pub}(A)}$

$\{N_b\}_{\text{pub}(I)} \rightarrow$

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

$\leftarrow \{N_a, N_b\}_{\text{pub}(A)}$

$\{N_b\}_{\text{pub}(B)} \rightarrow$

Agent A

Intruder I

Agent B

Attack

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

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

Exercise

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

Exercise

Propose a fix for the Needham-Schroeder protocol.