Hacka Academy
Introduction to Cryptography Security

Pascal Lafourcade

LIMOS

LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

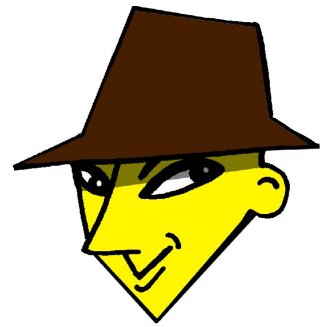
November 2018

Outline

Which adversary?










Adversary Model

Qualities of the adversary:

- Clever: Can perform all operations he wants
- Limited time:
  - Do not consider attack in 2^{60}.
  - Otherwise a Brute force by enumeration is always possible.

Model used: **Any Turing Machine**.

- Represents all possible algorithms.
- Probabilistic: adversary can generates keys, random number...

Adversary Models

The adversary is given access to oracles :

$\rightarrow$ encryption of all messages of his choice
$\rightarrow$ decryption of all messages of his choice

Three classical security levels:

- Chosen-Plain-text Attacks (CPA)

- Non adaptive Chosen-Cipher-text Attacks (CCA1)
  only before the challenge

- Adaptive Chosen-Cipher-text Attacks (CCA2)
  unlimited access to the oracle (except for the challenge)

Chosen-Plain-text Attacks (CPA)

Adversary can obtain all cipher-texts from any plain-texts.
It is always the case with a Public Encryption scheme.

Non adaptive Chosen-Cipher-text Attacks (CCA1)

Adversary knows the public key, has access to a **decryption oracle multiple times before to get the challenge** (cipher-text), also called "Lunchtime Attack" introduced by M. Naor and M. Yung ([NY90]).

Adaptive Chosen-Cipher-text Attacks (CCA2)

Adversary knows the public key, has access to a **decryption oracle multiple times before and AFTER to get the challenge**, but of course cannot decrypt the challenge (cipher-text) introduced by C. Rackoff and D. Simon ([RS92]).

Summary of Adversaries

CCA2: $\mathcal{O}_1 = \mathcal{O}_2 = \{\mathcal{D}\}$ Adaptive Chosen Cipher text Attack

$\Downarrow$

CCA1: $\mathcal{O}_1 = \{\mathcal{D}\}$, $\mathcal{O}_2 = \emptyset$ Non-adaptive Chosen Cipher-text Attack

$\Downarrow$

CPA: $\mathcal{O}_1 = \mathcal{O}_2 = \emptyset$ Chosen Plain text Attack

Outline

One-Wayness (OW)

Put your message in a translucid bag, but you cannot read the text.

One-Wayness (OW)

Put your message in a translucid bag, but you cannot read the text.

Without the private key, it is computationally **impossible to recover the plain-text**.

Is it secure ?

Is it secure ?

Is it secure ?

- you cannot read the text but you can distinguish which one has been encrypted.

Is it secure ?

- you cannot read the text but you can distinguish which one has been encrypted.
- Does not exclude to recover half of the plain-text
- Even worse if one has already partial information of the message:
  - Subject: XXXX
  - From: XXXX

Indistinguishability (IND)

Put your message in a black bag, you can not read anything.

Now a black bag is of course IND and it implies OW.

Indistinguishability (IND)

Put your message in a black bag, you can not read anything.

Now a black bag is of course IND and it implies OW.
The adversary is not able to **guess in polynomial-time even a bit of the plain-text knowing the cipher-text**, notion introduced by S. Goldwasser and S.Micali ([GM84]).

Is it secure?

Is it secure?

Is it secure?

- It is possible to scramble it in order to produce a new cipher. In more you know the relation between the two plain text because you know the moves you have done.

Non Malleability (NM)

Put your message in a black box.

But in a black box you cannot touch the cube (message), hence NM implies IND.

Non Malleability (NM)

Put your message in a black box.

But in a black box you cannot touch the cube (message), hence NM implies IND.
The adversary should **not be able to produce a new cipher-text** such that the plain-texts are meaningfully related, notion introduced by D. Dolev, C. Dwork and M. Naor in 1991 ([DDN91,BDPR98,BS99]).

Summary of Security Notions

Non Malleability

$\Downarrow$

Indistinguishability

$\Downarrow$

One-Wayness

Example: RSA

public	private
$n = pq$	$d = e^{-1} \mod \phi(n)$
e (public key)	(private key)

RSA Encryption

- $E(m) = m^e \mod n$
- $D(c) = c^d \mod n$

OW-CPA = RSA problem by definition!

But not semantically secure because it is deterministic.

Recall Elgamal

- $G = (\langle g \rangle, *)$ finite cyclic group of prime order q.
- x: private key.
- $y = g^x$: public key.

$\mathcal{E}(m; r) = (g^r, y^r m) \rightarrow (c, d)$ and $\mathcal{D}(c, d) = \frac{d}{c^x}$

OW = CDH Assumption
IND-CPA= DDH Assumption

OW-CPA for Elgamal

Exercice

Prove that under CDH assumption El-Gamal is OW-CPA.

OW-CPA for Elgamal

Exercice

Prove that under CDH assumption El-Gamal is OW-CPA.

Consider an adversary A that can invert random Elgamal encryptions. We will show that this quantity is negligible.
We first use A to build an adversary D for computing the Diffie-Hellman function:
D : Adversary to compute the Diffie-Hellman function:
On input (g^a, g^b), we must output g^{ab}.

1. Give g^a to A as the public key.
2. Pick a random $d \in G$ and give (g^b, d) to A as the ciphertext.
3. When A outputs $m = \frac{d}{g^{ab}}$, we output $\frac{d}{m}$.

IND-CPA for Elgamal

Exercice

Prove that under DDH assumption El-Gamal is IND-CPA.

IND-CCA2 for Cramer-Shoup

TODO

ECB Attack

Let us fix a block cipher $\mathcal{E} : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$. and $S\mathcal{E} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ an ECB symmetric encryption scheme, where the size of each block is n.

We build an adversary A with a high IND-CPA advantage.

$$\mathcal{E}_K(LR(m_l, m_r, b)) = \begin{cases} \mathcal{E}_K(m_l) & \text{if } b = 1 \\ \mathcal{E}_K(m_r) & \text{if } b = 0 \end{cases}$$

Adversary A

Adversary $A^{\mathcal{E}_K(LR(.,.,b)}$
$M_0 \leftarrow 0^n||1^n$;
$M_1 \leftarrow 0^{2n}$;
$C[1]C[2] \leftarrow \mathcal{E}_K(LR(M_0, M_1, b))$
If $C[1] = C[2]$ then return 1 else return 0

$X[i]$ denotes the i-th block of a string X, a block being a sequence of n bits.

Proof

$$Pr[Exp_{S\mathcal{E}}^{IND-CPA\ 0}(A) = 1] = 0$$

$$Pr[Exp_{S\mathcal{E}}^{IND-CPA\ 1}(A) = 1] = 1$$

Why?

Proof

$$Pr[Exp_{S\mathcal{E}}^{IND-CPA\ 0}(A) = 1] = 0$$

$$Pr[Exp_{S\mathcal{E}}^{IND-CPA\ 1}(A) = 1] = 1$$

Why?

- If $b = 1$, then the oracle returns $C[1]C[2] = \mathcal{E}_K(0^n)||\mathcal{E}_K(0^n)$, so $C[1] = C[2]$ and A returns 1.

Proof

$$Pr[Exp_{S\mathcal{E}}^{IND-CPA\ 0}(A) = 1] = 0$$

$$Pr[Exp_{S\mathcal{E}}^{IND-CPA\ 1}(A) = 1] = 1$$

Why?

- If $b = 1$, then the oracle returns $C[1]C[2] = \mathcal{E}_K(0^n)||\mathcal{E}_K(0^n)$, so $C[1] = C[2]$ and A returns 1.
- if $b = 0$, the oracle returns $C[1]C[2] = \mathcal{E}_K(0^n)||\mathcal{E}_K(1^n)$. Hence $C[1] \neq C[2]$. So A returns 0 in this case.

Proof

$$Pr[Exp_{S\mathcal{E}}^{IND-CPA\ 0}(A) = 1] = 0$$

$$Pr[Exp_{S\mathcal{E}}^{IND-CPA\ 1}(A) = 1] = 1$$

Why?

- If $b = 1$, then the oracle returns $C[1]C[2] = \mathcal{E}_K(0^n)||\mathcal{E}_K(0^n)$, so $C[1] = C[2]$ and A returns 1.
- if $b = 0$, the oracle returns $C[1]C[2] = \mathcal{E}_K(0^n)||\mathcal{E}_K(1^n)$. Hence $C[1] \neq C[2]$. So A returns 0 in this case.

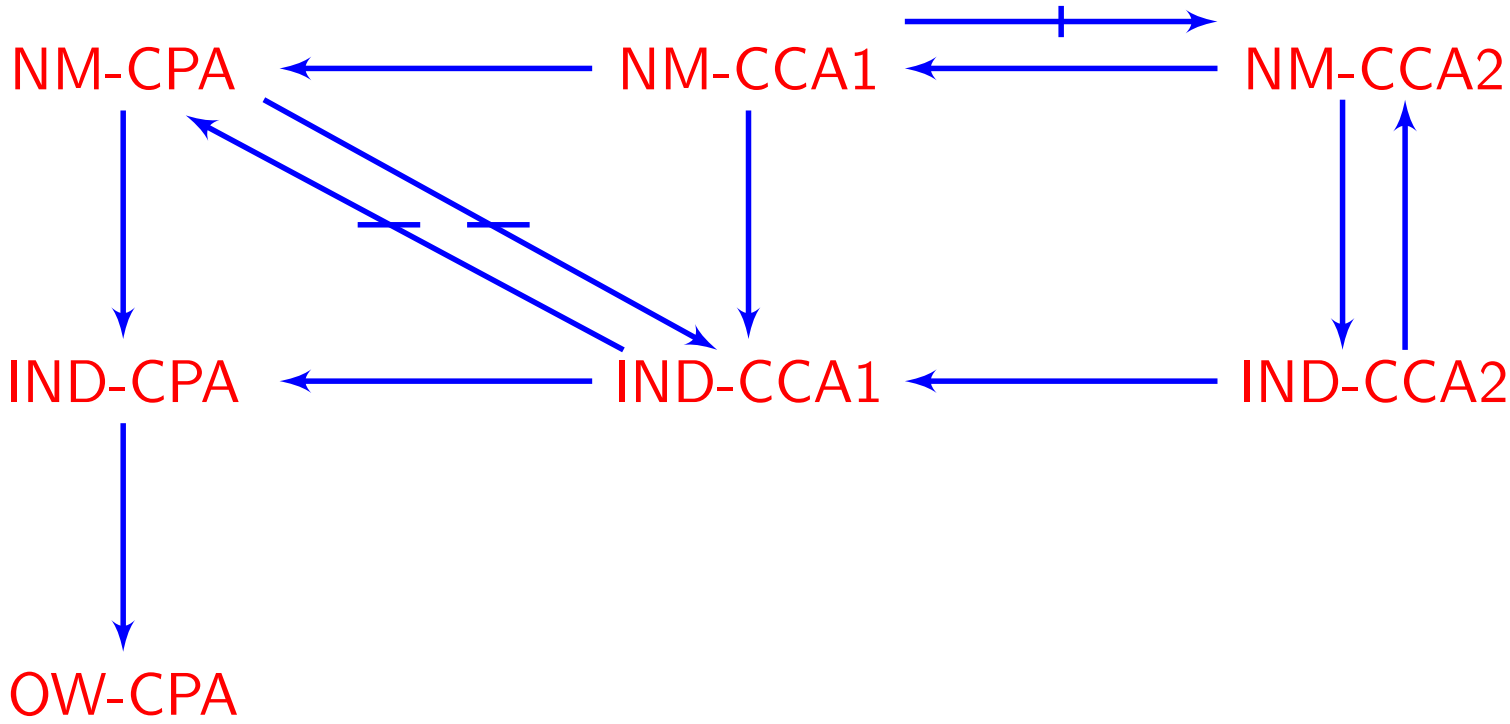
$$Adv_{S\mathcal{E}}^{IND-CPA}(A) = 1 - 0 = 1$$

This means that the ECB encryption scheme is insecure.

Exercise

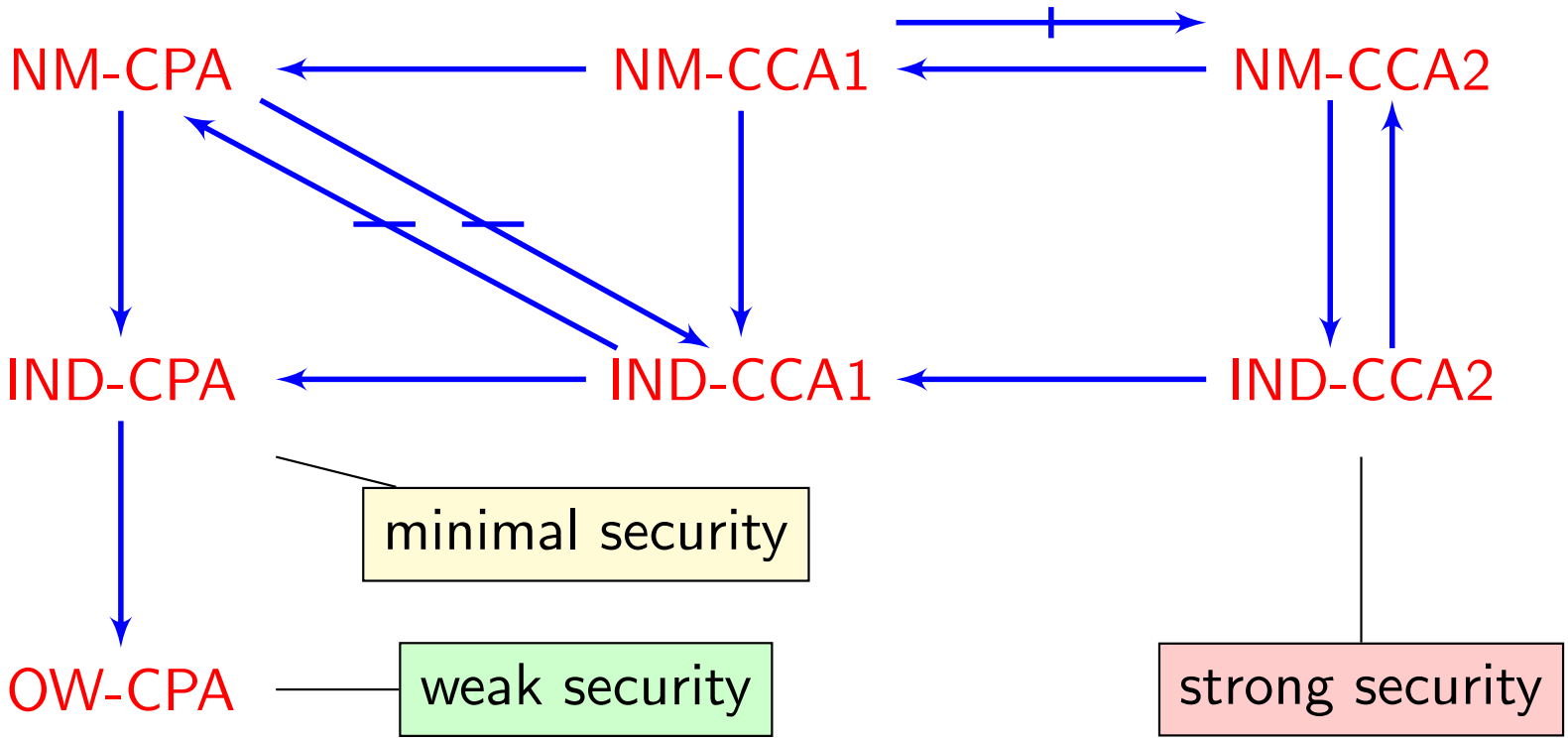
1. Find an attack on CBC with counter IV.
2. Prove that CBC with random IV is not IND-CCA2 secure.
3. Notice that CBC with random IV is IND-CPA secure.

Relations



"Relations Among Notions of Security for Public-Key Encryption Schemes", **Crypto'98**, by Mihir Bellare, Anand Desai, David Pointcheval and Phillip Rogaway [BDPR'98]

Relations



"Relations Among Notions of Security for Public-Key Encryption Schemes", **Crypto'98**, by Mihir Bellare, Anand Desai, David Pointcheval and Phillip Rogaway [BDPR'98]

Outline

Motivation for Security

- Communicaion email, conference
- Bank, e-commerce ...
- High-avaiblility

Limit of users education.
Changes are difficult and costly, might lose users' confidance.
(software backdoors, even by international organisations)
• How to do it securely even for novice ?

GOAL: novice user can achieve secure communication
(without a PhD)

Typical security-critical problems

- Secure communication, e.g., via telephone, email, fax.
  Objective: confidentiality and integrity of transmitted information.
- Internet banking. Objectives: confidentiality of transactions and account information, prevention of false transactions, impossibility of repudiating (denying) a transaction by a user, ...
- Digital payment systems.
- E-voting systems, ...
- Digital rights management.

N.B.: specifying objectives (security properties) is not always easy. Neither is building systems that satisfy these objectives!

Traditional security properties

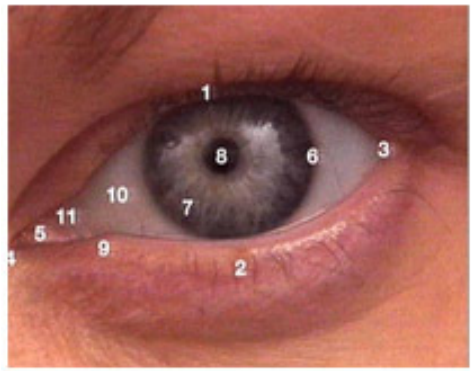
- Common security properties are:
  - Confidentiality or Secrecy: No improper disclosure of information
  - Authentification: To be sure to talk with the right person. disclosure of information
  - Integrity: No improper modification of information
  - Availability: No improper impairment of functionality/service

Authentication

"On the Internet, nobody knows you're a dog."

Mechanisms for Authentication

1. Something that you know
   E.g. a PIN or a password
2. Something that you have
   E.g. a smart-card



3. Something that you are
   Biometric characteristics like voice, fingerprints, eyes, ...
4. Where you are located
   E.g. in a secure building

Strong authentication combines multiple factors:
E.g., Smart-Card + PIN

Other security properties

- Non-repudiation (also called accountability) is where one can establish responsibility for actions.
- Fairness is the fact there is no advantage to play one role in a protocol comparing with the other ones.
- Privacy

  Anonymity: secrecy of principal identities or communication relationships.

  Pseudonymity: anonymity plus link-ability.

  Data protection: personal data is only used in certain ways.

Example: banking

- A bank may require
  - authenticity of clients (at teller, ATMs, or on the Internet),
  - non-repudiation of transactions,
  - integrity of accounts and other customer data,
  - secrecy of customer data, and
  - availability of logging.
- The conjunction of these properties might constitute the bank's (high-level) security policy.

Another example: e-voting

- An e-voting system should ensure that
  - only registered voters vote,
  - each voter can only vote once,
  - integrity of votes,
  - privacy of voting information (only used for tallying), and
  - availability of system during voting period
- In practice, many policy aspects are difficult to formulate precisely.

Exercise: Give the security properties that an international airport should guarantee.
More details with Florent Autreau later.

Outline

Example :

Boxes example:

- Alice puts lock on a box and sends it to Bob
- Bob adds a lock and sends it to Alice
- Alice removes her lock and sends back the lock to Bob
- Bob opens the box by removing his lock

Shamir 3-Pass Protocol

$$
\begin{array}{llcll}
1 & A & \rightarrow & B & : \quad \{m\}_{K_A} \\
2 & B & \rightarrow & A & : \quad \{\{m\}_{K_A}\}_{K_B} \{\{m\}_{K_B}\}_{K_A} \\
3 & A & \rightarrow & B & : \quad \{m\}_{K_B}
\end{array}
$$

Commutative Encryption

Two differents world

Computational World

- Find some bits of the key
- Find some bit of the message
- Cryptanalyse, key, message recovery
- Message = bits
- Intrduer = PPTTM

Computational World

- Message = term algebra
- Intruder = Deduction system
- Perfect encryption hypothesis
- Logical Attack

Simple Example

- DAY 1
  Alice sends to Bob $\{12h00\}_{K_B}$
- Day 2
  Alice sends to Bob $\{11h45\}_{K_B}$
  But intruder replays the message of DAY 1

This kind of attack is valid for all encryptions

Another Simple Example using RSA

$\{12345\}_{K_A}$
$\{\{12345\}_{K_A}\}_{K_B} = \{\{12345\}_{K_B}\}_{K_A}$
$\{12345\}_{K_B}$

$\{12345\}_{K_A}$
$\{\{12345\}_{K_A}\}_{K_I} = \{\{12345\}_{K_I}\}_{K_A}$
$\{12345\}_{K_I}$
Problem of Authentication

Examples of kinds of attack

- Man-in-the-middle (or parallel sessions) attack: pass messages through to another session $A \leftrightarrow I \leftrightarrow B$.
- Replay (or freshness) attack: record and later re-introduce a message or part.
- Reflection attack: send transmitted information back to originator.
- Oracle attack: take advantage of normal protocol responses as encryption and decryption "services".
- Type flaw (confusion) attack: substitute a different type of message field (e.g. a key vs. a name).

Messages Abstraction

- Names: A, B or Alice, Bob, ...
- Nonces: N_A. Fresh data.
- Keys: K and **inverse keys** K^{-1}
- Asymmetric Encryption: : $\{M\}_{K_A}$
- Symmetric Encryption: $\{M\}_{K_{AB}}$.
- Message concatenation: $\langle M_1, M_2 \rangle$.

Example: $\{\langle A \oplus N_B, K_{AB} \rangle\}_{K_B}$.

Example: Needham-Schroeder Protocol 1978

$\{A, N_A\}_{K_B}$
$\{N_A, N_B\}_{K_A}$
$\{N_B\}_{K_B}$

Question

- Is N_B a shared secret between A et B?

Exercise

Answer

- In 1995, G.Lowe find an attack 17 years after its publication!

Exercise: Try to Find an attack on this famous protocol.

If you already know this attack try to correct the protocol to avoid this attack.

If you already know the Lowe correction too try to find a flaw on this corrected protocol (Very Difficult)

Lowe Attack on the Needham-Schroeder

so-called "Man in the middle attack"

$$\{A, N_A\}_{K_I}$$
$$\{A, N_A\}_{K_B}$$
$$\{N_A, N_B\}_{K_A}$$
$$\{N_B\}_{K_I}$$
$$\{N_B\}_{K_B}$$

Needham-Schroeder corrected by Lowe 1995

$\{A, N_A\}_{K_B}$
$\{N_A, N_B, B\}_{K_A}$
$\{N_B\}_{K_B}$

Needham-Schroeder corrected by Lowe 1995

$\{A, N_A\}_{K_B}$
$\{N_A, N_B, B\}_{K_A}$
$\{N_B\}_{K_B}$

Needham-Schroeder corrected by Lowe 1995

$\{A, N_A\}_{K_B}$
$\{N_A, N_B, B\}_{K_A}$
$\{N_B\}_{K_B}$

Needham-Schroeder corrected by Lowe 1995

$\{A, N_A\}_{K_B}$
$\{N_A, N_B, B\}_{K_A}$
$\{N_B\}_{K_B}$

Question

- This time the protocol is secure?

Needham-Schroeder corrected by Lowe 1995

$\{A, N_A\}_{K_B}$
$\{N_A, N_B, B\}_{K_A}$
$\{N_B\}_{K_B}$

Question

- This time the protocol is secure?

Answer

- There exists a type flaw attack.

Type Flaw Attack on the Needham-Schroeder-Lowe

Recall

$\{A, N_A\}_{K_B}$

$\{N_A, N_B, B\}_{K_A}$

$\{N_B\}_{K_B}$

Attack

$\mathsf{I} \rightarrow \mathsf{B} : \{A, I\}_{K_B}$

$\mathsf{B} \rightarrow \mathsf{I} : \{I, N_b, B\}_{K_A}$

$\mathsf{I} \rightarrow \mathsf{A} : \{I, (N_b, B)\}_{K_A}$

$\mathsf{A} \rightarrow \mathsf{I} : \{(N_b, B), N_a, A\}_{K_I}$

$\mathsf{I} \rightarrow \mathsf{B} : \{N_b\}_{K_B}$

$\mathsf{I} \rightarrow \mathsf{A} : \{N_a\}_{K_A}$

Questions?

How can we find such attacks?

- Models for Protocols
- Models for Properties
- Theories
- Dedicated Techniques
- Tools
  - Automatic
  - Semi-automatic

Questions?

How can we find such attacks?

- Models for Protocols
- Models for Properties
- Theories
- Dedicated Techniques
- Tools
  - Automatic
  - Semi-automatic

It is the goal of this lecture

Why is it difficult to verify such protocols?

- Messages: Size not bounded
- Nonces: Arbitrary number
- Chanel: Unsecure
- Intruder: Unlimited capabilities
- Instances: Unbounded numbers of principals
- Interleaving: Unlimited applications of the protocol.

FFGG by J.Millen

1 $A \rightarrow B : A$

2 $B \rightarrow A : B, N, M$

3 $A \rightarrow B : A, \{N, M, S\}_{PkB}$ for B view $A, \{N, X, S\}_{PkB}$

4 $B \rightarrow A : N, X, \{X, S, N\}_{PkB}$

FFGG by J.Millen

1 $A \rightarrow B : A$

2 $B \rightarrow A : B, N, M$

3 $A \rightarrow B : A, \{N, M, S\}_{PkB}$ for B view $A, \{N, X, S\}_{PkB}$

4 $B \rightarrow A : N, X, \{X, S, N\}_{PkB}$

Is S secret ?

FFGG by J.Millen

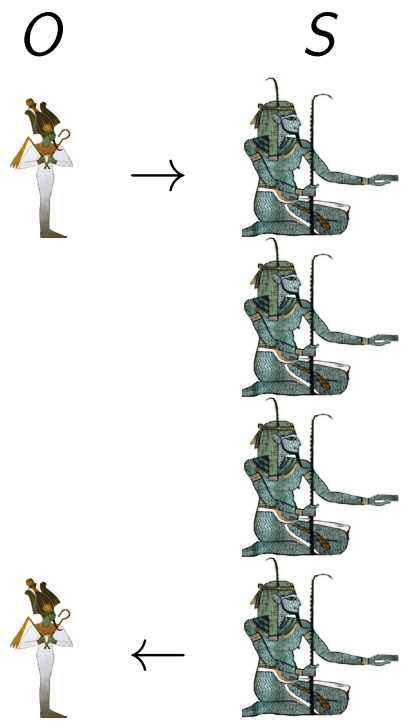
1 $A \rightarrow B : A$
2 $B \rightarrow A : B, N, M$
3 $A \rightarrow B : A, \{N, M, S\}_{PkB}$ for B view $A, \{N, X, S\}_{PkB}$
4 $B \rightarrow A : N, X, \{X, S, N\}_{PkB}$
Is S secret ?

Parallel Attack

1.1	$A \rightarrow B$	:	A	
2.1	$A \rightarrow B$	:	A	
1.2	$B \rightarrow I(A)$	:	B, N_1, M_1	
2.2	$B \rightarrow I(A)$	:	B, N_2, M_2	
1.2	$I(B) \rightarrow A$	:	B, N_1, N_2	a
1.3	$A \rightarrow B$	:	$A, \{N_1, N_2, S\}_{PkB}$	b
1.4	$B \rightarrow A$	:	$N_1, N_2, \{N_2, S, N_1\}_{PkB}$	c
2.3	$I(A) \rightarrow B$	:	$A, \{N_2, S, N_1\}_{PkB}$	d
2.4	$B \rightarrow A$	:	$N_2, S, \{S, N_1, N_2\}_{PkB}$	

TMN Protocol: Distribution of a fresh symmetric key

[Tatebayashi, Matsuzuki, Newmann 89]:



$$O \rightarrow S : O, I, \{N_O\}_{Pub_S}$$

$$S \rightarrow I : S, O$$

$$I \leftarrow S : I, O, \{N_I\}_{Pub_S}$$

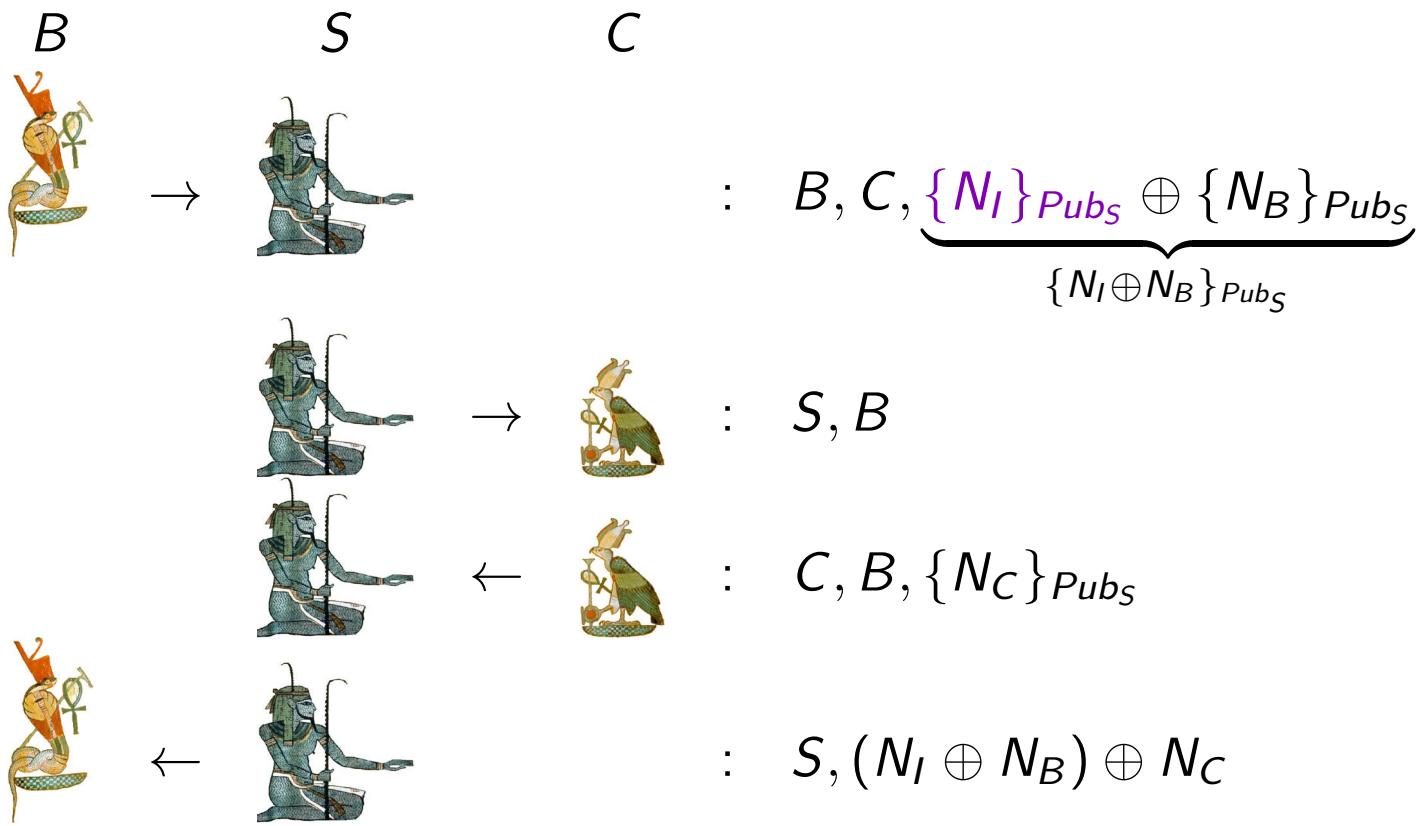
$$O \leftarrow S : S, I, N_O \oplus N_I$$

Osiris retrieves N_I:

Using $x \oplus x \oplus y = y$, knowing N_O

Attack on TMN Protocol [Simmons'94]

With homomorphic encryption $\{a\}_k \oplus \{b\}_k = \{a \oplus b\}_k$



Buto Learns:

Using $x \oplus x \oplus y = y$, knowing N_B and N_C, he deduces N_I.

Paper

Prudent Engineering Practice for Cryptographic Protocols
Martín Abadi and Roger Needham
November 1, 1995

Principle 1

"**Every message should say what it means:** the interpretation of the message should depend only on its content. It should be possible to write down a straightforward English sentence describing the content—though if there is a suitable formalism available that is good too."

A recipient can recover the meaning without any context.

Principle 2

"The conditions for a message to be acted upon should be clearly set out so that someone reviewing a design may see whether they are acceptable or not."

Example: Attack on Wide-mouthed-frog protocol

Wide Moth Frog

Protocol by Michael Burrows (1989)

1. A $\rightarrow$ S : A, {Ta, B, Kab}Kas
2. S $\rightarrow$ B : {Ts, A, Kab}Kbs

Attack Lowe 1997

- i.1. A $\rightarrow$ S : A, {Ta, B, Kab}Kas
- i.2. S $\rightarrow$ B : {Ts, A, Kab}Kbs
- ii.2. S $\rightarrow$ B : {Ts, A, Kab}Kbs

Principle 3

"If the identity of a principal is essential to the meaning of a message, it is prudent to mention the principal's name explicitly in the message.

Several examples TODO

Denning Sacco

1. $A \rightarrow S : \quad \langle A, B \rangle$
2. $S \rightarrow A : \quad \{\langle\langle B, K_{AB}\rangle, \langle N_s, \{\langle K_{AB}, \langle A, N_s\rangle\rangle\}_{K_{BS}}\rangle\rangle\}_{K_{AS}}$
3. $A \rightarrow B : \quad \{\langle K_{AB}, \langle A, N_s\rangle\rangle\}_{K_{BS}}$
4. $B \rightarrow A : \quad \{S_{AB}\}_{K_{AB}}$

Attack Lowe

$i.1. \quad A \rightarrow S : \quad A, B$

$i.2. \quad S \rightarrow A : \quad \{B, Kab, N_S, \{Kab, A, N_S\}Kbs\}Kas$

$i.3. \quad A \rightarrow B : \quad \{Kab, A, N_S\}Kbs$

$ii.3. \quad I(A) \rightarrow B : \quad \{Kab, A, N_S\}Kbs$

In session ii, B thinks that A wants to establish a new shared key and accepts it.

Principle 4

"Be clear about why encryption is being done. Encryption is not wholly cheap, and not asking precisely why it is being done can lead to redundancy. Encryption is not synonymous with security, and its improper use can lead to errors."

Examples : Needham Schroeder + TLS todo

Principle 5

" When a principal signs material that has already been encrypted, it should not be inferred that the principal knows the content of the message. On the other hand, it is proper to infer that the principal that signs a message and then encrypts it for privacy knows the content of the message."

Example : The CCITT X.509 standard TODO

Principle 6

"Be clear what properties you are assuming about nonces. What may do for ensuring temporal succession may not do for ensuring association—and perhaps association is best established by other means."

Example : Otway-Rees 87 and toWoo and Lam (todo)

Otway-Rees 1987

Protocol

1. $A \rightarrow B : M, A, B, \{Na, M, A, B\}Kas$
2. $B \rightarrow S : M, A, B, \{Na, M, A, B\}Kas , \{Nb, M, A, B\}Kbs$
3. $S \rightarrow B : M, \{Na, Kab\}Kas, \{Nb, Kab\}Kbs$
4. $B \rightarrow A : M, \{Na, Kab\}Kas$

Attack : John Clark and Jeremy Jacob 1997

- $A \rightarrow I(B) : M, A, B, \{Na, M, A, B\}Kas$
- $B \rightarrow S : M, A, B, \{Na, M, A, B\}Kas , \{Nb, M, A, B\}Kbs$
- $S \rightarrow B : M, \{Na, Kab\}Kas, \{Nb, Kab\}Kbs$
- $I(B) \rightarrow A : M, \{Na, M, A, B\}Kas$

Principle 7

"The use of a predictable quantity (such as the value of a counter) can serve in guaranteeing newness, through a challenge-response exchange. But if a predictable quantity is to be effective, it should be protected so that an intruder cannot simulate a chal- lenge and later replay a response."

Example

1. $A \rightarrow S : A; Na$
2. $S \rightarrow A : \{Ts; Na\}Kas$

corrected version : When Na is predictable, it should be protected

1. $A \rightarrow S : A; \{Na\}Kas$
2. $S \rightarrow A : \{Ts; \{Na\}Kas\}Kas$

Principle 8

"If timestamps are used as freshness guarantees by reference to absolute time, then the difference between local clocks at various machines must be much less than the allowable age of a message deemed to be valid. Furthermore, the time maintenance mechanism everywhere becomes part of the trusted computing base"

Example: Gong

Principle 9

"A key may have been used recently, for example to encrypt a nonce, yet be quite old, and possibly compromised. Recent use does not make the key look any better than it would otherwise."

NS + Varadharajan, Allen, and Black TODO

- $A \rightarrow S : A; B; Na$
- $S \rightarrow A : \{Na; B; Kab; \{Kab; A\}Kbs\}Kas$
- $A \rightarrow B : \{Kab; A\}Kbs$
- $B \rightarrow A : \{Nb\}Kab$
- $A \rightarrow B : \{Nb + 1\}Kab$

Principle 10

"If an encoding is used to present the meaning of a message, then it should be possible to tell which encoding is being used. In the common case where the encoding is protocol dependent, it should be possible to deduce that the message belongs to this protocol, and in fact to a particular run of the protocol, and to know its number in the protocol

TODO examples

Principle 11

"The protocol designer should know which trust relations his protocol depends on, and why the dependence is necessary. The reasons for particular trust relations being acceptable should be explicit though they will be founded on judgment and policy rather than on logic. "

TODO examples

Outline

The Diffie-Hellman protocol

g, p are public parameters.

- Diffie chooses x and computes $g^x \mod p$
- Diffie sends $g^x \mod p$
- Hellman chooses y and computes $g^y \mod p$
- Hellman sends $g^x \mod p$

Shared key: $(g^x)^y = g^{xy} = (g^y)^x$
Basic Diffie-Hellman key-exchange: initiator I and responder R exchange public "half-keys" to arrive at mutual session key $\mathrm{k} = \mathrm{g}^{\mathrm{xy}} \mod \mathrm{p}$.

Hard Problems

Most cryptographic constructions are based on *hard problems*.
Their security is proved by reduction to these problems:

- RSA. Given $N = pq$ and $e \in \mathbb{Z}^*_{\varphi(N)}$, compute the inverse of e modulo $\varphi(N) = (p-1)(q-1)$. Factorization
- Discrete Logarithm problem, DL. Given a group $\langle g \rangle$ and g^x, compute x.
- Computational Diffie-Hellman, CDH Given a group $\langle g \rangle$, g^x and g^y, compute g^{xy}.
- Decisional Diffie-Hellman, DDH Given a group $\langle g \rangle$, distinguish between the distributions (g^x, g^y, g^{xy}) and (g^x, g^y, g^r).

The Discrete Logarithm (DL)

Let $G = (\langle g \rangle, *)$ be any finite cyclic group of prime order.

Idea: it is hard for any adversary to produce x if he only knows g^x.
For any adversary $\mathcal{A}$,

$$\mathbf{Adv}^{DL}(\mathcal{A}) = Pr\left[\mathcal{A}(g^x) \rightarrow x \middle| x, y \xleftarrow{R} [1, q]\right]$$

is negligible.

Computational Diffie-Hellman (CDH)

Idea: it is hard for any adversary to produce g^{xy} if he only knows g^x and g^y.
For any adversary $\mathcal{A}$,

$$\mathbf{Adv}^{CDH}(\mathcal{A}) = Pr\left[\mathcal{A}(g^x, g^y) \rightarrow g^{xy} \middle| x, y \xleftarrow{R} [1, q]\right]$$

is negligible.

Decisional Diffie-Hellman (DDH)

Idea: Knowing g^x and g^y, it should be hard for any adversary to distinguish between g^{xy} and g^r for some random value r.
For any adversary $\mathcal{A}$, the advantage of $\mathcal{A}$

$$\mathbf{Adv}^{DDH}(\mathcal{A}) = Pr\left[\mathcal{A}(g^x, g^y, g^{xy}) \rightarrow 1 \middle| x, y \xleftarrow{R} [1, q]\right]$$
$$- Pr\left[\mathcal{A}(g^x, g^y, g^r) \rightarrow 1 \middle| x, y, r \xleftarrow{R} [1, q]\right]$$

is negligible.
This means that an adversary cannot extract a single bit of information on g^{xy} from g^x and g^y.

Relation between the problems

Prop

Solve DL $\Rightarrow$ Solve CDH $\Rightarrow$ Solve DDH. (Exercise)

Prop (Moaurer & Wolf)

For many groups, DL $\Leftrightarrow$ CDH

Prop (Joux & Wolf)

There are groups for which DDH is easier than CDH.

Proofs by Reduction

Solve DL $\Rightarrow$ Solve CDH

Attack on DL implies attack on CDH.
Given g, g^x, g^y using DL we get x and y so we can compute g^{xy}.

Solve CDH $\Rightarrow$ Solve DDH

Attack on CDH implies attack on DDH.
Given g, g^x, g^y, g^r using CDH we compute g^{xy} and we can compare with g^r.

Usage of DH assumption

The Diffie-Hellman problems are widely used in cryptography:

- Public key crypto-systems [ElGamal, Cramer& Shoup]
- Pseudo-random functions [Noar& Reingold, Canetti]
- Pseudo-random generators [Blum& Micali]
- (Group) key exchange protocols [many]

The Diffie-Hellman protocol

g, p are public parameters.

The Diffie-Hellman protocol

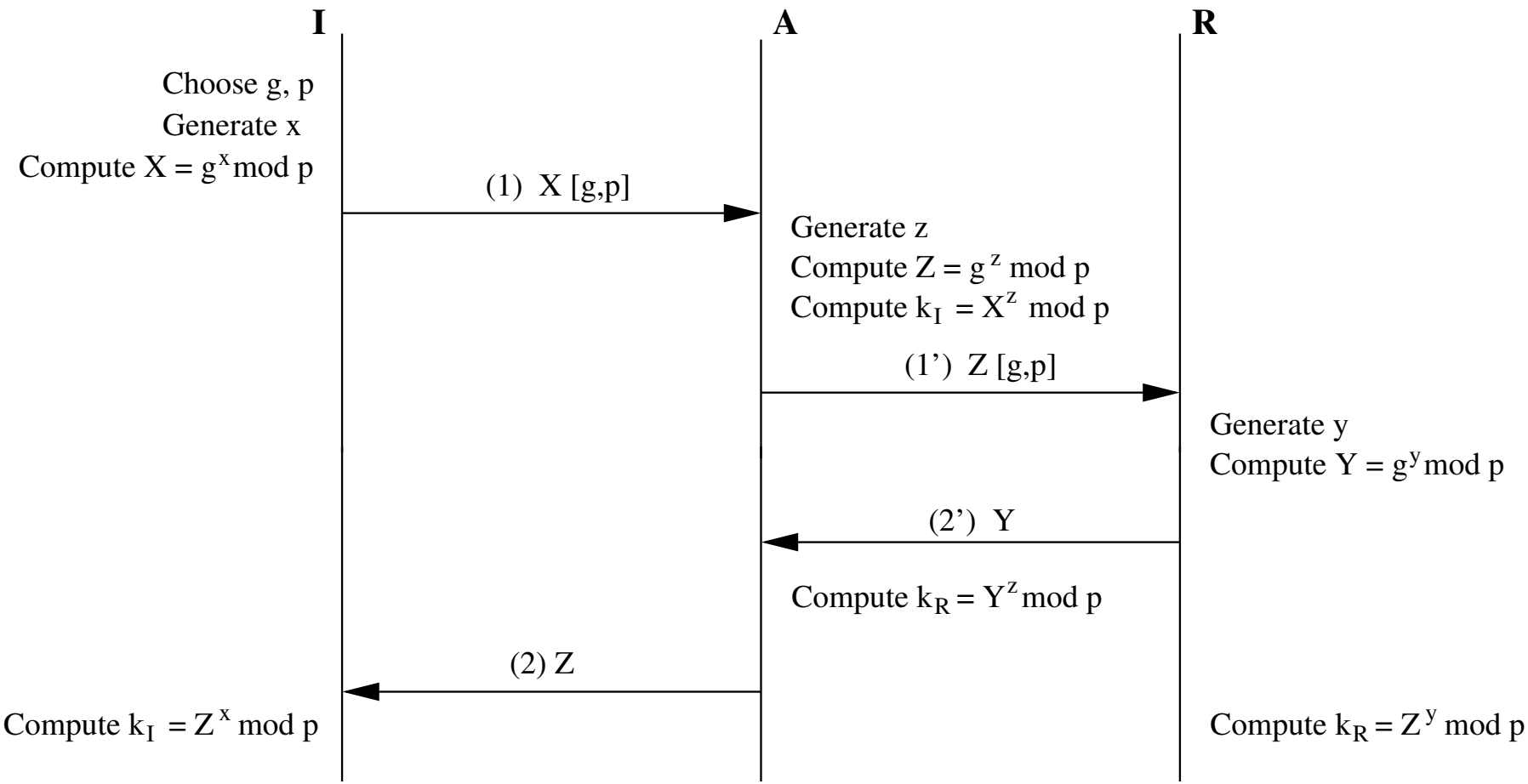
g, p are public parameters.

The Diffie-Hellman protocol

g, p are public parameters.

$$(g^y)^x \bmod p = k = g^{xy} \bmod p = (g^x)^y \bmod p$$

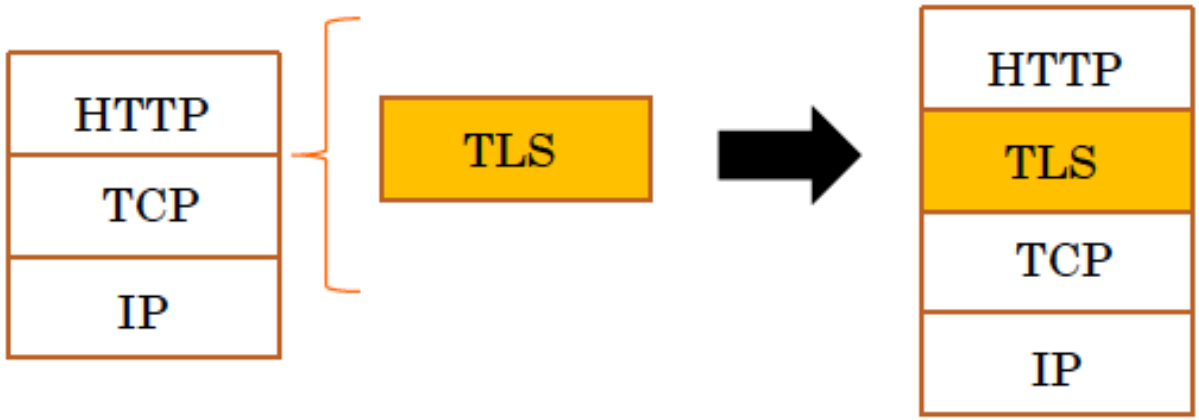
Man-in-the-middle attack



Outline

Le protocole SSL/TLS

- Protocole de *sécurisation des échanges* sur Internet
- Mode *client-serveur*, au dessus de *TCP*



Le protocole SSL/TLS

- Permet de garantir
  - l'*authentification* du serveur
  - la *confidentialité* des données échangées
  - l'*intégrité* et l'*authentification de l'origine* des données échangées
  - l'*authentification* du client (optionnel)
- Permet d'encapsuler de manière transparente des protocoles de la *couche application*
  - HTTP (80) $\rightarrow$ HTTP*S* (443)
  - IMAP (143) $\rightarrow$ IMAP*S* (993)
  - POP3 (110) $\rightarrow$ POP3*S* (995)
  - *STARTTLS* (pour IMAP, POP3, SMTP, FTP, etc.) sur le *même port*

Un peu d'histoire

- Au début, *SSL* (*Secure Sockets Layer*), développé par Netscape
  - SSL 1.0 (1994) : protocole théorique, jamais utilisé
  - SSL 2.0 (1995-2011) : première version utilisée
  - SSL 3.0 (1996-..., RFC 6101) : dernière version, sur laquelle TLS sera basé
- Puis *TLS* (*Transport Layer Security*), développé par l'IETF (*Internet Engineering Task Force*)
  - TLS 1.0 (1999-..., RFC 2246) : successeur de SSL, diverses amélioration jusqu'en 2002
  - TLS 1.1 (2006-..., RFC 4346)
  - TLS 1.2 (2008-..., RFC 5246)
  - TLS 1.3 (2017 -..., Under discussion)
- *Compatibilité* des serveurs HTTPS (source : SSL Pulse, mars 2016)

SSL 2.0	SSL 3.0	TLS 1.0	TLS 1.1	TLS 1.2
8,4 %	26,8 %	98,2 %	71,9 %	74,2 %

Mécanismes cryptographiques dans TLS

- TLS offre le choix entre *plusieurs mécanismes cryptographiques* pour être capable de s'adapter aux *contraintes* de chaque application et de chaque système
- *Authentification* du serveur (et du client, optionnellement) :
  - clé publique : RSA, DSS, ECDSA
  - clé secrète ou mot de passe partagé : PSK (*Pre-Shared Key*), SRP (*Secure Remote Password*)
  - pas d'authentification : ANON
- *Échange de clés* :
  - clé publique (toujours la même clé privée côté serveur) : RSA
  - Diffie-Hellman statique (même problème) : DH, ECDH
  - clé secrète ou mot de passe partagé : PSK, SRP
  - Diffie-Hellman éphémère (clés privées *propres à chaque connexion* ; garantit la *forward secrecy*) : DHE, ECDHE

Mécanismes cryptographiques dans TLS

- *Chiffrement* :
  - chiffrement par bloc : AES-CBC, 3DES-CBC, DES-CBC, etc.
  - chiffrement par bloc avec mode authentifiant : AES-CCM, AES-GCM, etc.
  - chiffrement par flot : RC4
  - pas de chiffrement : NULL
- *Intégrité* et *authentification de l'origine* des messages :
  - HMAC : HMAC-MD5, HMAC-SHA1, HMAC-SHA256, etc.
  - mode authentifiant du chiffrement par bloc : AEAD
- Une combinaison de ces mécanismes est appelée *cipher suite*
  - par exemple : TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- En bonus, TLS peut aussi supporter un *mode de compression* des données : NULL ou DEFLATE

Quelques chiffres sur SSL/TLS

- plus de 50 RFC
- 5 versions à ce jour
- plus de 300 suites cryptographiques
- plus de 20 extensions
- des fonctionnalités intéressantes: compression, renégociation, reprise de session (2 méthodes), une dizaine d'implémentations connues, mais combien d'implémentations maison ?

Implementation

Quelle réponse peut attendre un client proposant les suites cryptographiques suivantes : AES128-SHA et ECDH-ECDSA-AES128-SHA ?

- A: AES128-SHA
- B: ECDH-ECDSA-AES128-SHA
- C: une alerte
- D: la réponse RC4-MD5

Une suite cryptographique est représentée par un entier sur 16 bits.
Pendant longtemps, toutes les suites étaient de la forme 00 XX
Pourquoi perdre du temps à regarder l'octet de poids fort
Code pour RC4-MD5 = 00 05
Code pour ECDH-ECDSA-AES128-SHA = C0 05

Client Hello 258 octets

Un ClientHello TLS dont la taille est comprise entre 256 et 511 peut être confondu avec un ClientHello SSLv2 !

16	03	01	01	02

TLS

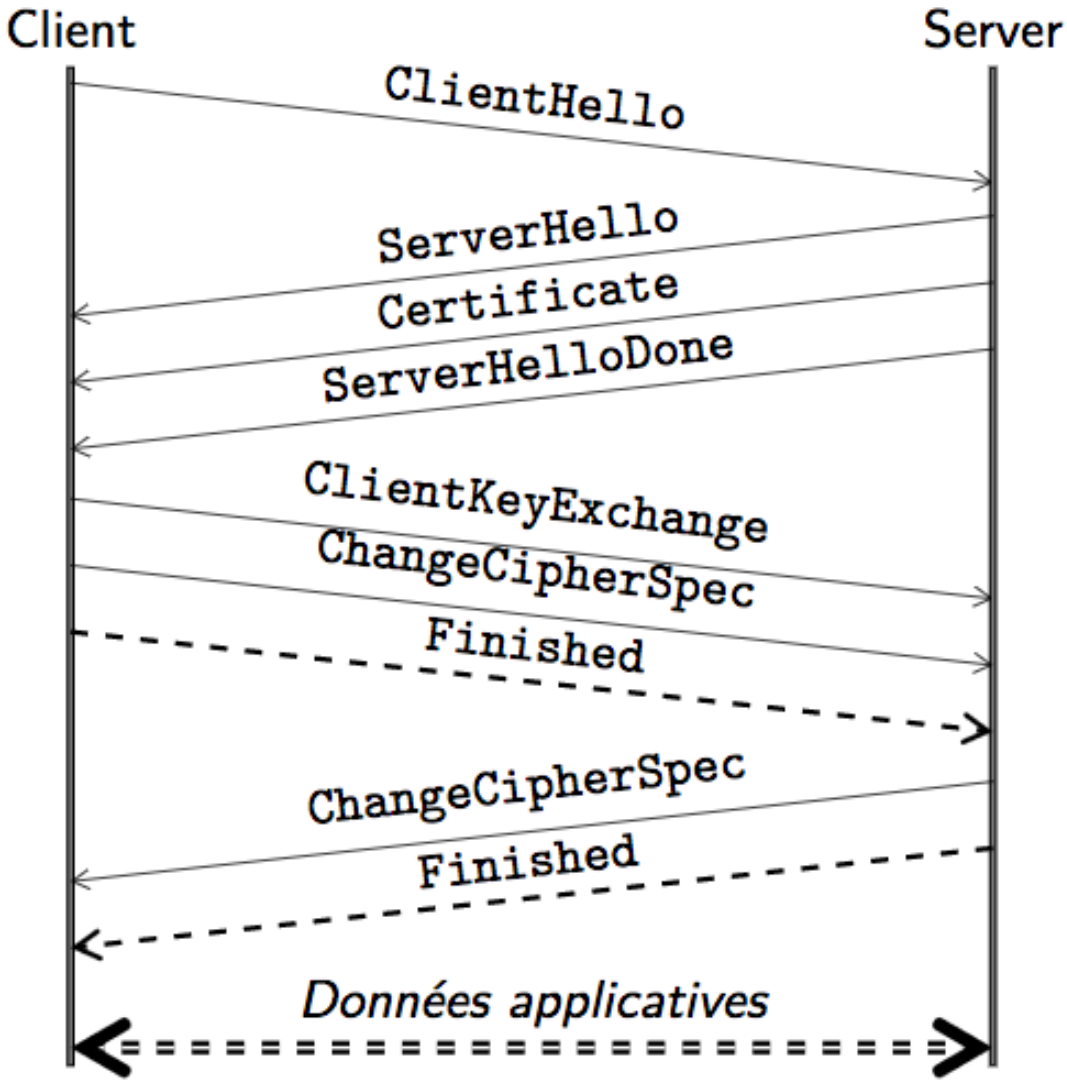
Type	Version	longueur
HS	TLS 1.0	258

SSLv2

longueur	Pad	Type
5635	...	CH

01 correspond à un ClientHello ainsi 1603 devient 5635 octets !

High level


Client
Server
ClientHello
ServerHello
Certificate
ServerHelloDone
ClientKeyExchange
ChangeCipherSpec
Finished
ChangeCipherSpec
Finished
Données applicatives

TLS

TLS = 5 protocoles

- *Handshake*, connexion sécurisée entre le client et le serveur
- *Change Cipher Spec*, nouvelle clef de session va être utilisée
- *Alert* $\Rightarrow$ Warning ou Fatal
- *Application Data*, encapsulation des données après Handshake
- *TLS Record*, encapsule puis relaye les données (Chiffremennt symétrique et MAC)

Établissement d'une connexion SSL/TLS

- Contexte : un *client* souhaite établir une connexion SSL/TLS avec un *serveur*
- Objectifs :
  - se mettre d'accord sur une *cipher suite commune*
  - *authentifier* le serveur
  - échanger des *clés secrètes* pour le chiffrement et l'authentification des messages
- Protocole de *handshake* en 4 étapes

Handshake SSL/TLS simple

1. *Client* $\rightarrow$ *Serveur*
   - `ClientHello` : plus haute version du protocole supportée, liste des *cipher suites* et modes de compression supportés
2. *Serveur* $\rightarrow$ *Client*
   - `ServerHello` : version du protocole, *cipher suite* et mode de compression choisis
   - `Certificate` (opt.) : la clé publique du serveur dans un certificat X.509 permettant d'en vérifier l'authenticité
   - `ServerKeyExchange` (opt.) : une clé publique Diffie-Hellman pour l'échange de clés
   - `ServerHelloDone`

Handshake SSL/TLS simple

3. *Client* → *Serveur*
   - `ClientKeyExchange` : soit un secret (*PreMasterKey*) chiffré avec la clé publique du serveur, soit une clé publique Diffie-Hellman pour l'échange de clés
   - `ChangeCipherSpec` (marque la fin du *handshake* côté client)
   - `Finished` : message chiffré et authentifié contenant un MAC des messages précédents du *handshake*
4. *Serveur* → *Client* (si le `Finished` du client est valide)
   - `ChangeCipherSpec` (marque la fin du *handshake* côté serveur)
   - `Finished` : message chiffré et authentifié contenant un MAC des messages précédents du *handshake*
5. Si le `Finished` du serveur est aussi valide, la connexion est établie

Authentification du serveur

- Dans le *handshake* précédent, le serveur envoie *lui-même* sa *clé publique* afin d'*authentifier ses messages*
  → N'y a-t-il pas comme un problème ?
- Comment *vérifier l'authenticité de la clé publique* du serveur ?
  - Clé publique *signée par le serveur* ?
    → La vérification nécessite de faire confiance à la clé publique...
  - Clé publique *signée par une tierce partie* (appelée *autorité de certification*, ou CA) ?
    → OK, mais comment vérifier cette signature ?
  - Clé publique *signée par un CA*, et *clé publique du CA* ?
    → Comment vérifier l'*authenticité de la clé publique du CA* ?
    On a juste déplacé le problème...
- Il s'agit d'un problème difficile, qui nécessite la mise en place d'une *infrastructure à clés publiques* (ou PKI)

Infrastructure à clés publiques

- Permet de répondre à la question :

  Comment faire confiance à une clé publique ?

- Certificat : *signature de la clé publique* par un tiers de confiance
- Infrastructures possibles :
  - hiérarchique : *autorités de certification* (SSL/TLS et X.509, EMV)
  - décentralisée : *réseau de confiance* (PGP, GnuPG)

Autorités de certification

- Différents niveaux :
  - *CA racines* : peuvent certifier les *clés publiques d'autres CA*
  - *CA intermédiaires* : en général, *ne peuvent pas* certifier d'autres CA
    $\rightarrow$ certifient les *clés publiques des serveurs*
- Chaîne de certification :

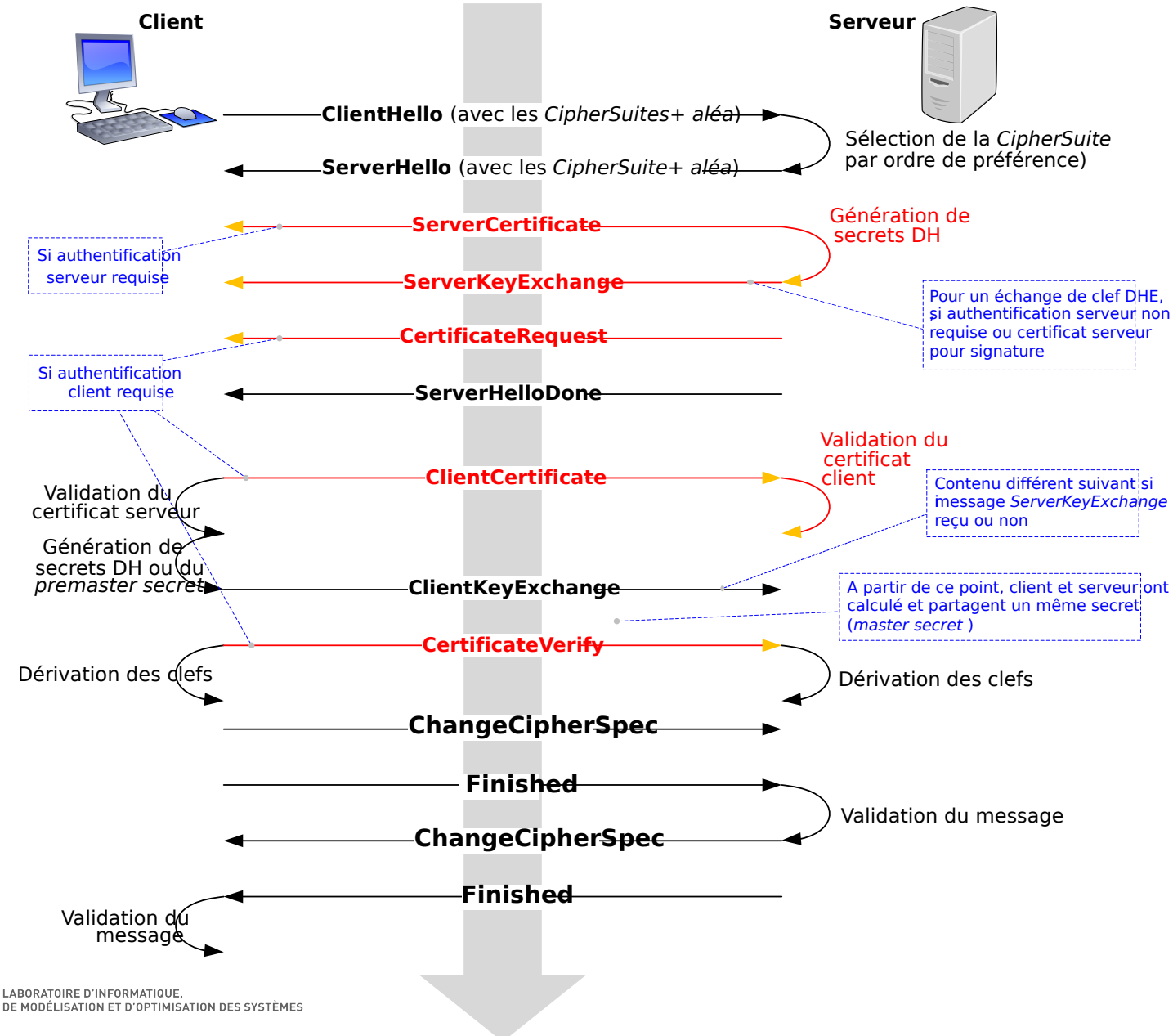
$$\textit{CA racine} \xrightarrow{\text{signe}} \textit{CA 1} \xrightarrow{\text{signe}} \textit{CA 2} \xrightarrow{\text{signe}} \textit{Serveur}$$

- *Authentification* : le serveur envoie trois certificats
  - sa propre *clé publique*, *signée* par CA 2
  - la *clé publique* de CA 2, *signée* par CA 1
  - la *clé publique* de CA 1, *signée* par CA racine
- *Vérification* : si le client connaît (et fait confiance à) la *clé publique de CA racine*, il peut vérifier la validité de *tous les certificats*

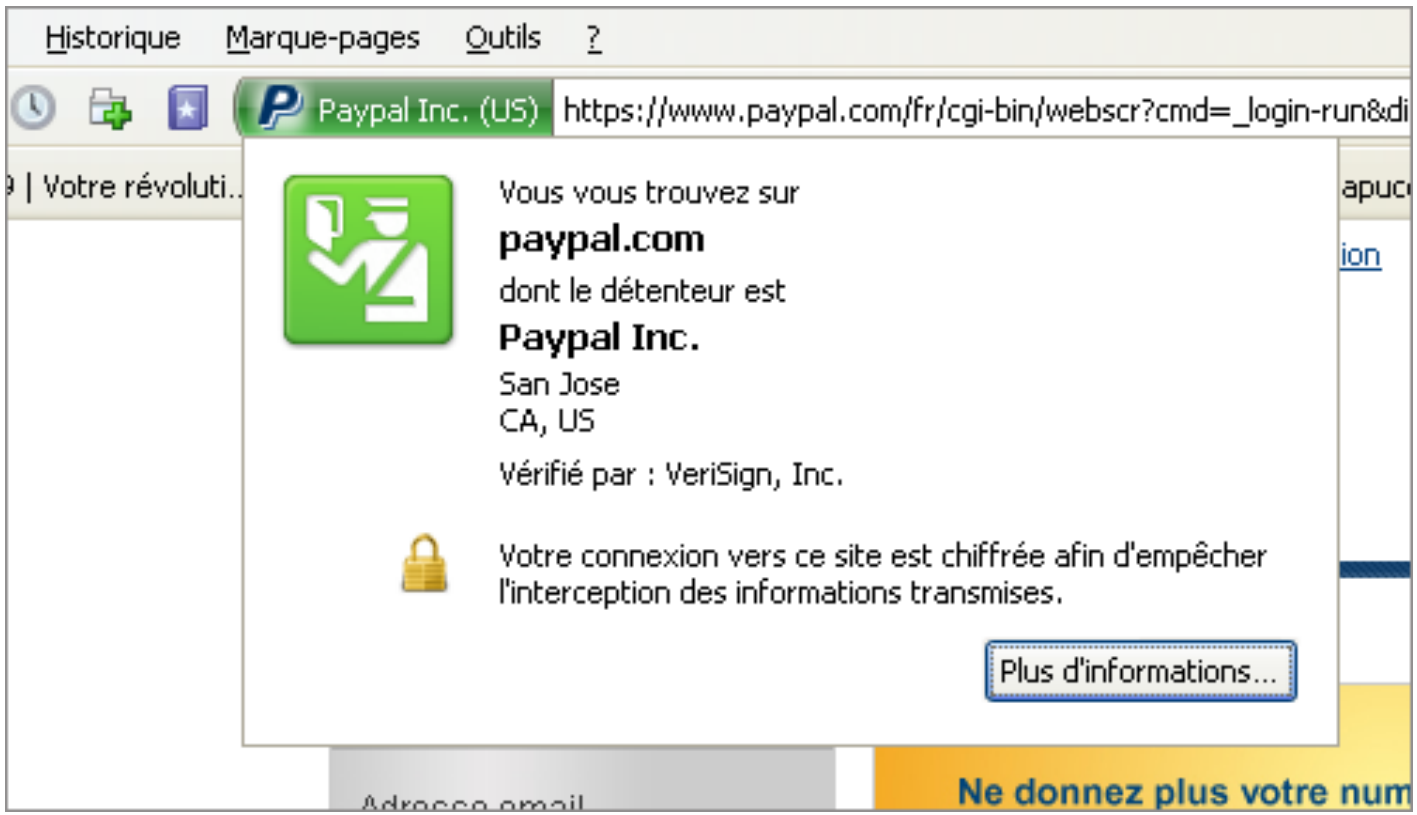
Autorités de certification

- Toujours les mêmes problèmes :
  - comment obtenir la *clé publique des CA racines* ?
  - *quelle confiance* leur accorder ?
- Liste de *CA racines de confiance* maintenue par les *navigateurs*
  - actuellement, 80+ *CA racines* intégrés dans Firefox
  - mais comment *avoir confiance en le navigateur* que l'on télécharge ?!
    $\rightarrow$ *contrôle d'intégrité et d'authenticité* de l'archive téléchargée ?
    $\rightarrow$ *problème de la poule et de l'œuf*...
- *Attention à la sécurité des CA* (racines et intermédiaires) !
  - si la *clé privée* d'un CA est compromise : *émission de faux certificats* p.ex. *DigiNotar* en 2011 : 500 faux certificats, dont `*.google.com`
  - *audits de sécurité* réguliers
  - *mécanisme de révocation* de certificats compromis : CRL (*Certificate Revocation List*) ou OCSP (*Online Certificate Status Protocol*)

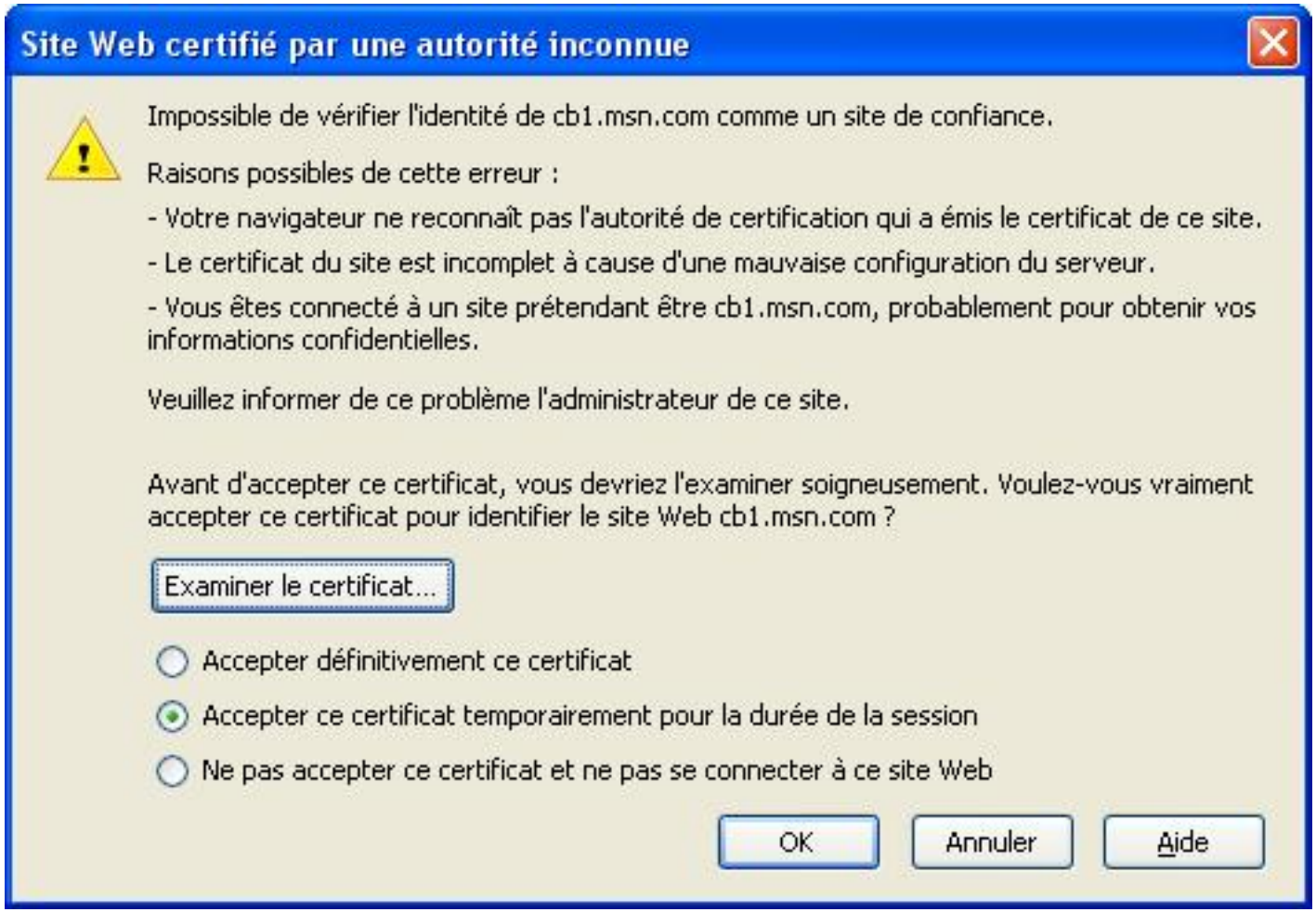
Application : TLS Handshake



Application : TLS



Application :



AC est inconnue du magasin de certificats.

Application :

Cette connexion n'est pas certifiée

Vous avez demandé à Iceweasel de se connecter de manière sécurisée à **static.ak.facebook.com**, mais nous ne pouvons pas confirmer que votre connexion est sécurisée.

Normalement, lorsque vous essayez de vous connecter de manière sécurisée, les sites présentent une identification certifiée pour prouver que vous vous trouvez à la bonne adresse. Cependant, l'identité de ce site ne peut pas être vérifiée.

Que dois-je faire ?

Si vous vous connectez habituellement à ce site sans problème, cette erreur peut signifier que quelqu'un essaie d'usurper l'identité de ce site et vous ne devriez pas continuer.

Sortir d'ici !

Détails techniques

static.ak.facebook.com utilise un certificat de sécurité invalide.

Le certificat n'est valide que pour les noms suivants :
a248.e.akamai.net , *.akamaihd.net , *.akamaihd-staging.net

(Code d'erreur : ssl_error_bad_cert_domain)

Je comprends les risques

- Soit le site Web est faux
- Soit des certificats distincts sont créés pour des sites distincts
- Soit il faut ajouter une valeur dans un champ

Outline

TLS 1.2 Handshake (AKE)

Client (C): Pick N_C and KE_C
C $\rightarrow$ S : N_C, ciphers, ext
Server (S): Pick N_S and has (PK_S, SK_S)
S $\rightarrow$ C : N_s, $Cert_S$, ciphers, ext
where $Cert_S = \{S, PK_S\}_{CA}$
Client checks $Cert_S$, computes pmk
$msk = HMAC(pmk; N_C||N_S)$
$K_C||K_S = HMAC(msk; 0||N_C||N_S)$
$Fin_c = HMAC(msk; 1||\tau)$ where τ is the transcript of previous messages
C $\rightarrow$ S : $KE_C||\{Fin_c\}_{K_C}$
S computes pmk, msk, $K_C||K_S$, checks Fin_c, computes
$Fin_S = HMAC(msk; 2||\tau)$
S $\rightarrow$ C : $\{Fin_S\}_{K_S}$
Checks Fin_s
How to compute pre-master key (pmk) ? 3 modes

TLS 1.2 RSA for computing pre-master key (pmk)

Most used.
Client (C): Pick N_c and KE_C
C $\rightarrow$ S : N_C
Server (S): Pick N_s and has (PK_S, SK_S) (RSA Key)
S $\rightarrow$ C : N_s, $Cert_S$
Client checks $Cert_S$, choose $pmk \in_R \{0,1\}^{8*48}$
$KE_C = RSA_{PK_S}(pmk)$
C $\rightarrow$ S : KE_C
S finds pmk by decrypting with SK_S associated to PK_S.

TLS 1.2 DH for computing pre-master key (pmk)

Client (C): Pick N_c
C $\rightarrow$ S : N_C
Server (S): Picks N_s and $KE_S = g^{ke_S} \mod p$ (DH Public Key)
S $\rightarrow$ C : N_s, $Cert(KE_S)$, KE_S
Client checks $Cert(KE_S)$, choose $ke_C \in_R \{0, q-1\}$
$KE_C = g^{ke_C} \mod p$, $pmk = KE_S^{ke_C} \mod p$
C $\rightarrow$ S : KE_C
S computes $pmk = KE_C^{ke_s} \mod p$.

TLS 1.2 DHE (Ephemeral) for pre-master key (pmk)

Client (C): Pick N_c
C $\rightarrow$ S : N_C
Server (S): Picks N_s and $KE_S = g^{ke_S} \mod p$ (Fresh DH Public Key)
S $\rightarrow$ C : N_s, G, $Cert(KE_S)$, KE_S
Client checks $Cert(KE_S)$, choose $ke_C \in_R \{0, q-1\}$
$KE_C = g^{ke_C} \mod p$, $pmk = KE_S^{ke_C} \mod p$
C $\rightarrow$ S : KE_C
S computes $pmk = KE_C^{ke_s} \mod p$.

Key recognition

Runs of TLs are sessions and have session IDs.

- If Client has seen before. reuse key material (msk)
- Use sID instead of N_C and N_S
  $C \rightarrow S : N_C, sID$
  $S \rightarrow C : N_S, sID, \{Fin_S\}_{K_S}$
  $K_C||K_S = HMAC(msk_{sID}; 0||N_C||N_S)$
  $Fin_c = HMAC(msk_{sID}; 1||\tau)$ where τ is the transcript of previous messages
  $C \rightarrow S : \{Fin_C\}_{K_C}$

Summary

Session Freshness

- Nonces N_C and N_S involved in key derivation
- Prevent replay attacks

Server Authentication

- Certificate enseures only server shares key with client
- Unilateral : anyone can exchange keys with server

Key Confirmation

- Last message: auhtenticated encryptuon with session keys
- Both parties are sure they computed the same keys

TLS 1.2 Some problems

- Configuration parameter not part of key
- Compatibility of ciphers and size not verified

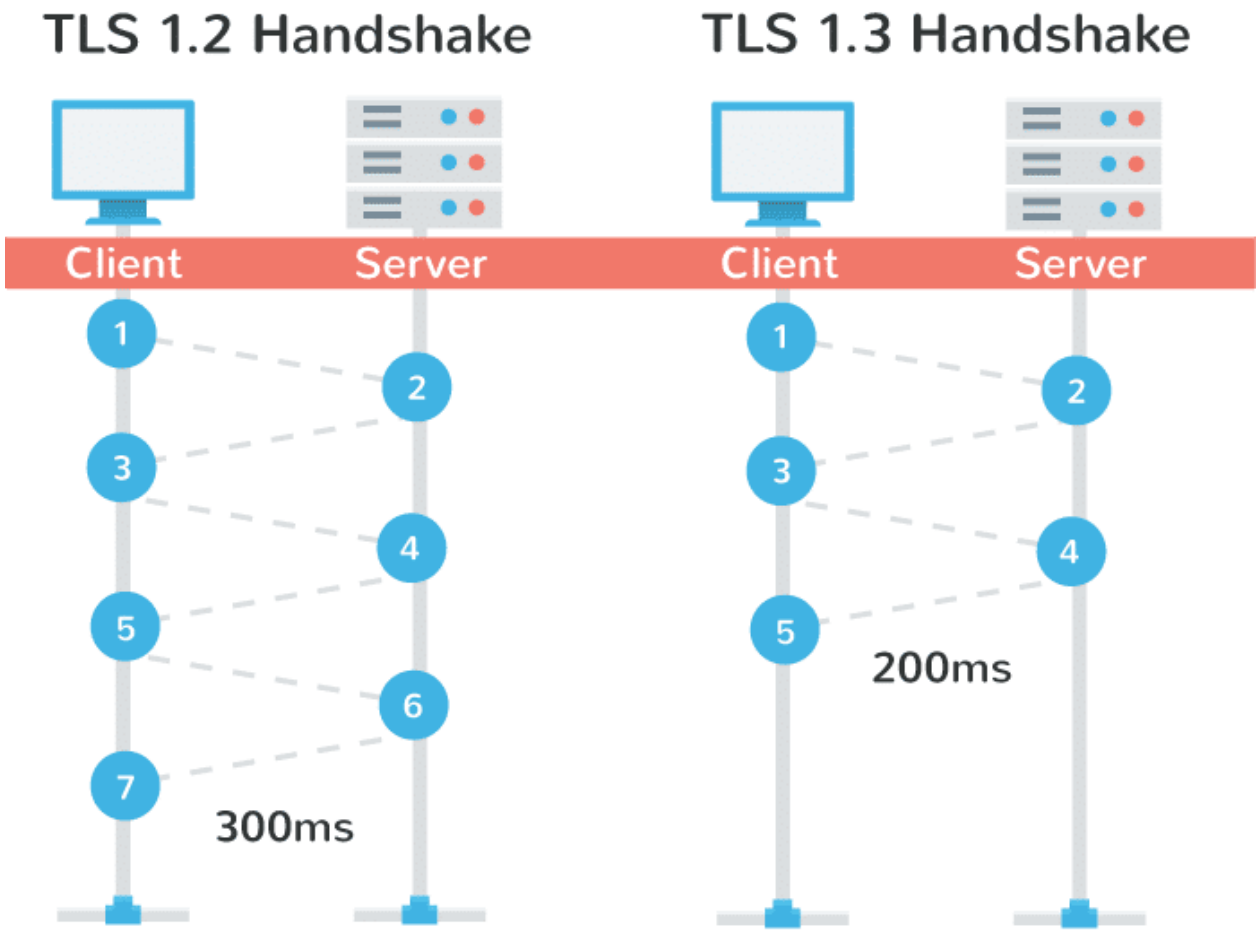
Some assmuptions:

- msk is computed with a Truly random function.
- Key expansion function is a PRF
- Gap DH is hard

TLS is secure proof in 2013, 2014.

Outline

Aperçu


TLS 1.2 Handshake
Client
Server
1
2
3
4
5
6
7
300ms
TLS 1.3 Handshake
Client
Server
1
2
3
4
5
200ms

TLS 1.3

- Clean up: Remove unused or unsafe features
- Security: Improve security by using modern security analysis techniques
- Privacy: Encrypt more of the protocol
- Performance: Our target is a 1-RTT handshake for naive clients; 0-RTT handshake for repeat connections
- Continuity: Maintain existing important use cases

`https://tlswg.github.io/tls13-spec/`

TLS 1.3 removes obsolete and insecure features

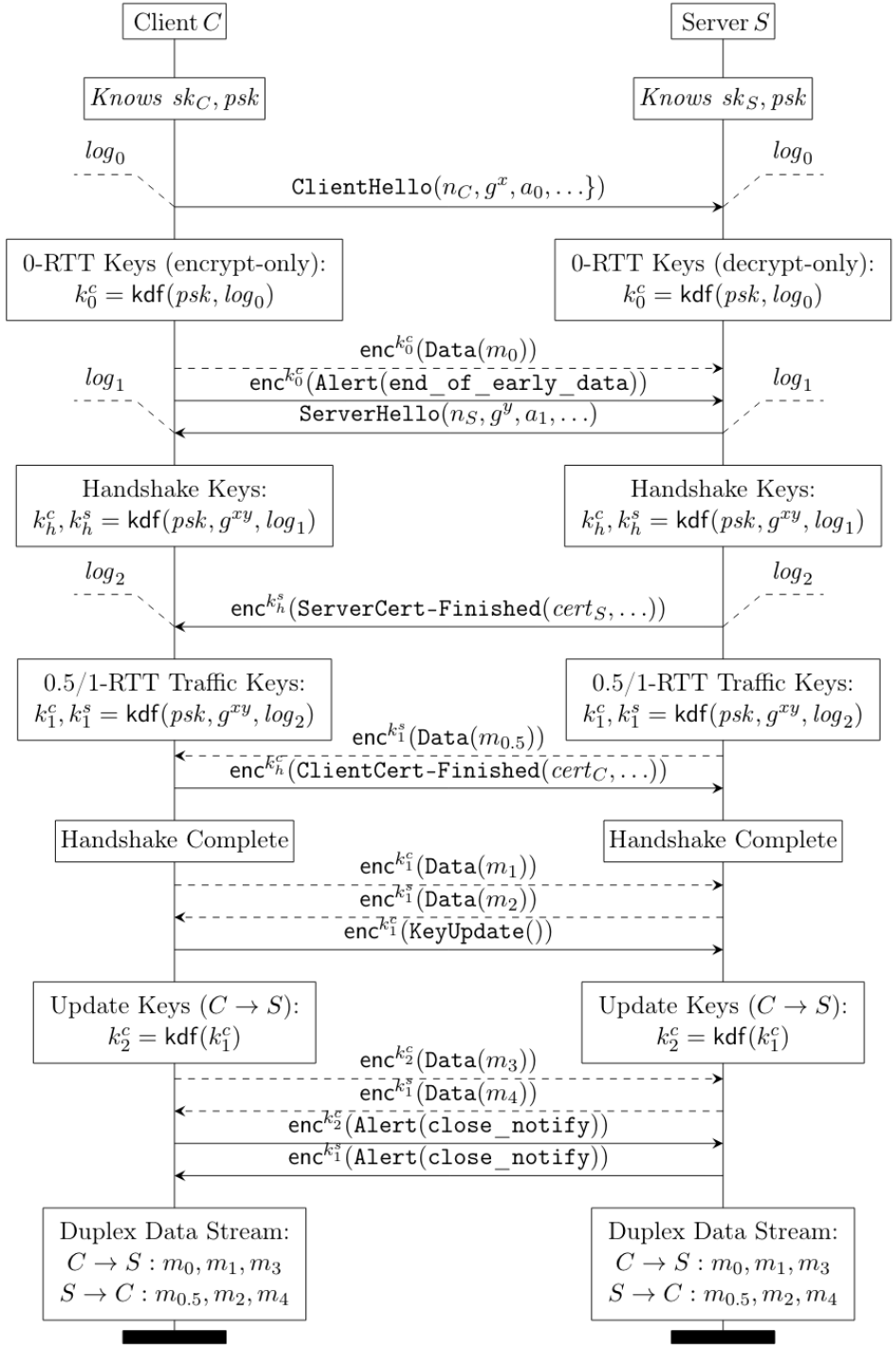
- SHA-1
- RC4
- DES
- 3DES
- AES-CBC
- MD5
- Arbitrary Diffie-Hellman groups — CVE-2016-0701
- EXPORT-strength ciphers – Responsible for FREAK and LogJam

TLS 1.3 1-RTT handshake: 12 messages in 3 flights, 16 derived keys, then data exchange.

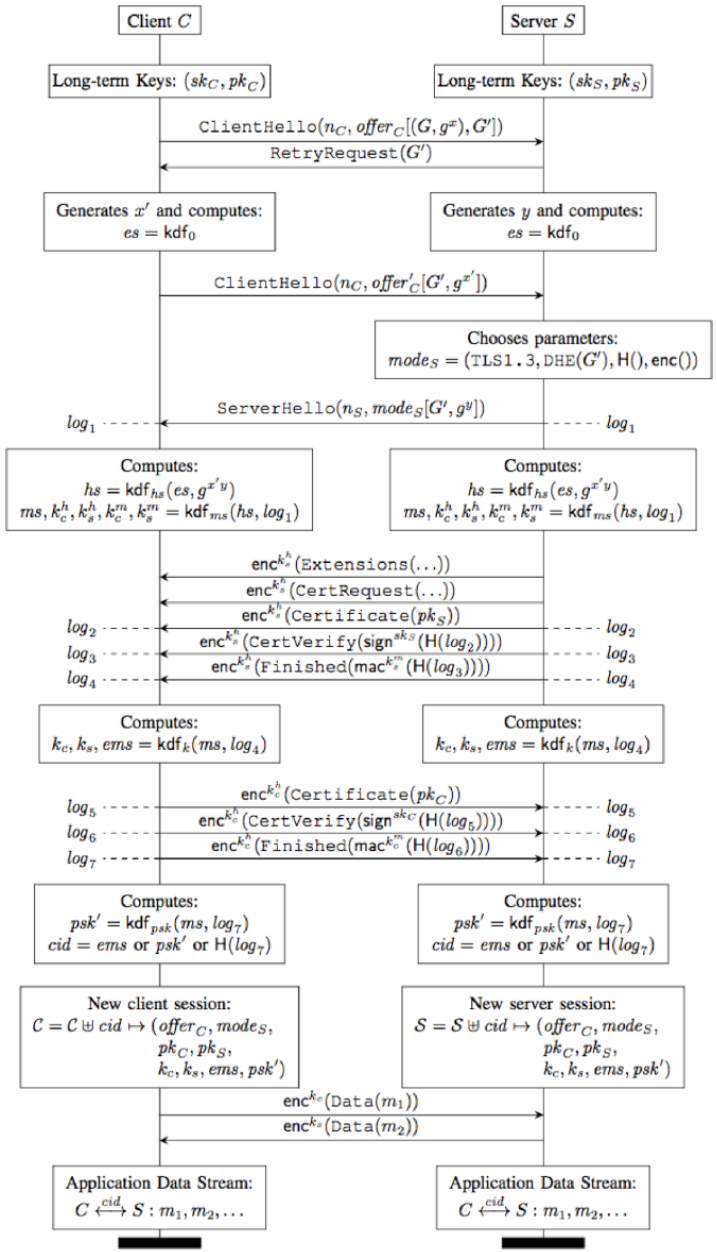
TLS 1.3

$C \rightarrow S : CHello, PK_C$, where $CHello = N_C||ciphers||ext$, supported by C
$S \rightarrow C : SHello, PK_S$ where $CHello = N_S||ciphers||ext||sub$, supported by S include in $CHello$.
$S \rightarrow C : \{Cert_S\}_{htk}$
$S \rightarrow C : \{CVf\}_{htk}$
$S \rightarrow C : \{Fin_S\}_{htk}$
$C \rightarrow S : \{Fin_C\}_{htk}$

TLS 1.3


Client C
Server S
Knows sk_C, psk
Knows sk_S, psk
log_0
log_0
$\texttt{ClientHello}(n_C, g^x, a_0, \ldots\})$
0-RTT Keys (encrypt-only):
$k_0^c = \texttt{kdf}(psk, log_0)$
0-RTT Keys (decrypt-only):
$k_0^c = \texttt{kdf}(psk, log_0)$
$\texttt{enc}^{k_0^c}(\texttt{Data}(m_0))$
log_1
$\texttt{enc}^{k_0^c}(\texttt{Alert}(\texttt{end\_of\_early\_data}))$
log_1
$\texttt{ServerHello}(n_S, g^y, a_1, \ldots)$
Handshake Keys:
$k_h^c, k_h^s = \texttt{kdf}(psk, g^{xy}, log_1)$
Handshake Keys:
$k_h^c, k_h^s = \texttt{kdf}(psk, g^{xy}, log_1)$
log_2
log_2
$\texttt{enc}^{k_h^s}(\texttt{ServerCert-Finished}(cert_S, \ldots))$
0.5/1-RTT Traffic Keys:
$k_1^c, k_1^s = \texttt{kdf}(psk, g^{xy}, log_2)$
0.5/1-RTT Traffic Keys:
$k_1^c, k_1^s = \texttt{kdf}(psk, g^{xy}, log_2)$
$\texttt{enc}^{k_1^s}(\texttt{Data}(m_{0.5}))$
$\texttt{enc}^{k_h^c}(\texttt{ClientCert-Finished}(cert_C, \ldots))$
Handshake Complete
Handshake Complete
$\texttt{enc}^{k_1^c}(\texttt{Data}(m_1))$
$\texttt{enc}^{k_1^s}(\texttt{Data}(m_2))$
$\texttt{enc}^{k_1^c}(\texttt{KeyUpdate}())$
Update Keys ($C \to S$):
$k_2^c = \texttt{kdf}(k_1^c)$
Update Keys ($C \to S$):
$k_2^c = \texttt{kdf}(k_1^c)$
$\texttt{enc}^{k_2^c}(\texttt{Data}(m_3))$
$\texttt{enc}^{k_1^s}(\texttt{Data}(m_4))$
$\texttt{enc}^{k_2^c}(\texttt{Alert}(\texttt{close\_notify}))$
$\texttt{enc}^{k_1^s}(\texttt{Alert}(\texttt{close\_notify}))$
Duplex Data Stream:
$C \to S : m_0, m_1, m_3$
$S \to C : m_{0.5}, m_2, m_4$
Duplex Data Stream:
$C \to S : m_0, m_1, m_3$
$S \to C : m_{0.5}, m_2, m_4$

TLS 1.3



Key Derivation Functions:

$\mathsf{hkdf\text{-}extract}(k, s) = \mathsf{HMAC\text{-}H}^k(s)$

$\mathsf{hkdf\text{-}expand\text{-}label}_1(s, l, h) =$
$\quad \mathsf{HMAC\text{-}H}^s(len_{\mathsf{H}()} \| \text{“}\texttt{TLS 1.3,}\text{”} \| l \| h \| \texttt{0x01})$

$\mathsf{derive\text{-}secret}(s, l, m) = \mathsf{hkdf\text{-}expand\text{-}label}_1(s, l, \mathsf{H}(m))$

1-RTT Key Schedule:

$\mathsf{kdf}_0 = \mathsf{hkdf\text{-}extract}(0^{len_{\mathsf{H}()}}, 0^{len_{\mathsf{H}()}})$

$\mathsf{kdf}_{hs}(es, e) = \mathsf{hkdf\text{-}extract}(es, e)$

$\mathsf{kdf}_{ms}(hs, log_1) = ms, k_c^h, k_s^h, k_c^m, k_s^m$ where
$\quad ms = \mathsf{hkdf\text{-}extract}(hs, 0^{len_{\mathsf{H}()}})$
$\quad hts_c = \mathsf{derive\text{-}secret}(hs, \mathsf{hts}_c, log_1)$
$\quad hts_s = \mathsf{derive\text{-}secret}(hs, \mathsf{hts}_s, log_1)$
$\quad k_c^h = \mathsf{hkdf\text{-}expand\text{-}label}(hts_c, \mathsf{key}, \text{“”})$
$\quad k_c^m = \mathsf{hkdf\text{-}expand\text{-}label}(hts_c, \mathsf{finished}, \text{“”})$
$\quad k_s^h = \mathsf{hkdf\text{-}expand\text{-}label}(hts_s, \mathsf{key}, \text{“”})$
$\quad k_s^m = \mathsf{hkdf\text{-}expand\text{-}label}(hts_s, \mathsf{finished}, \text{“”})$

$\mathsf{kdf}_k(ms, log_4) = k_c, k_s, ems$ where
$\quad ats_c = \mathsf{derive\text{-}secret}(ms, \mathsf{ats}_c, log_4)$
$\quad ats_s = \mathsf{derive\text{-}secret}(ms, \mathsf{ats}_s, log_4)$
$\quad ems = \mathsf{derive\text{-}secret}(ms, \mathsf{ems}, log_4)$
$\quad k_c = \mathsf{hkdf\text{-}expand\text{-}label}(ats_c, \mathsf{key}, \text{“”})$
$\quad k_s = \mathsf{hkdf\text{-}expand\text{-}label}(ats_s, \mathsf{key}, \text{“”})$

$\mathsf{kdf}_{psk}(ms, log_7) = psk'$ where
$\quad psk' = \mathsf{derive\text{-}secret}(ms, \mathsf{rms}, log_7)$

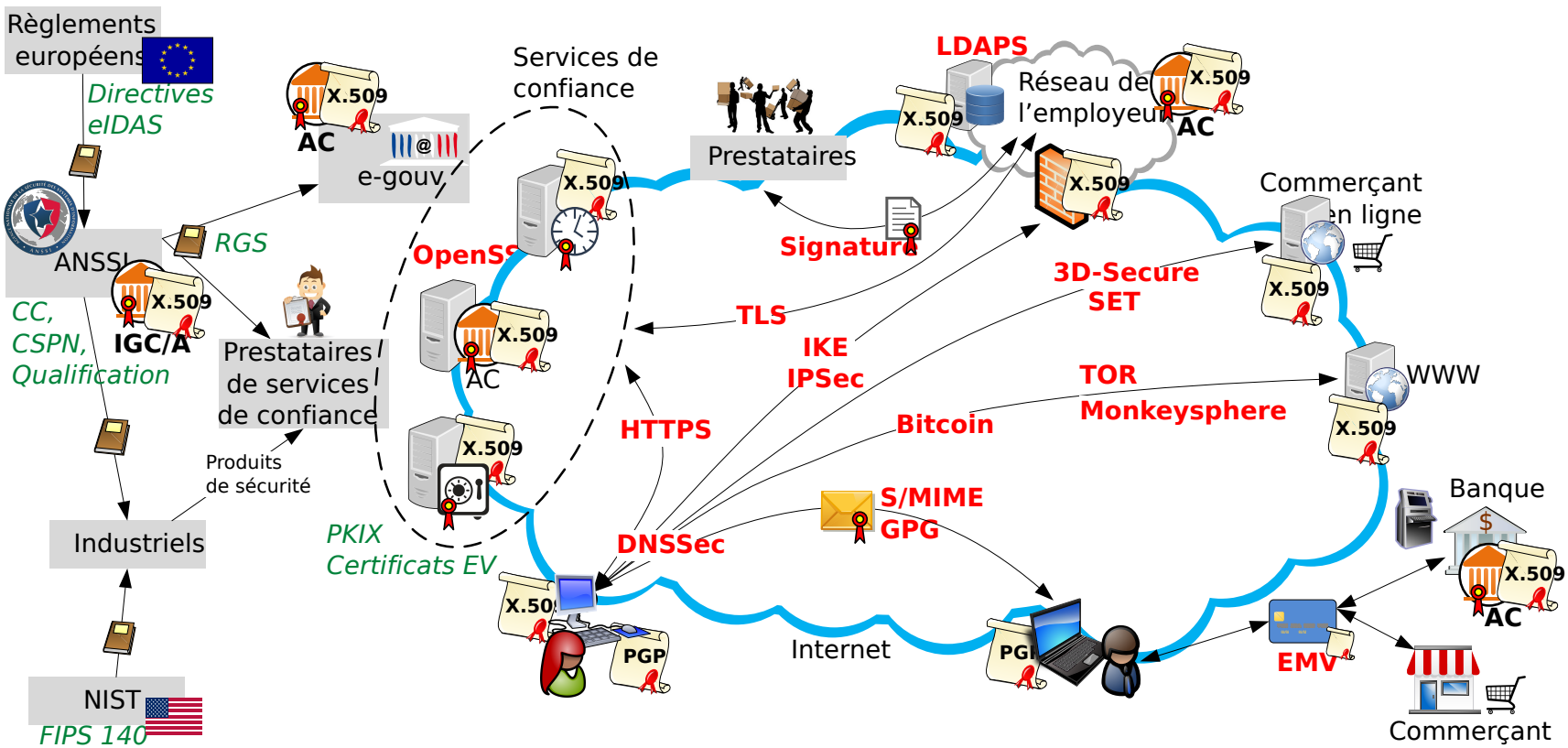
PSK-based Key Schedule:

$\mathsf{kdf}_{es}(psk) = es, k^b$ where
$\quad es = \mathsf{hkdf\text{-}extract}(0^{len_{\mathsf{H}()}}, psk)$
$\quad k^b = \mathsf{derive\text{-}secret}(es, \mathsf{pbk}, \text{“”})$

$\mathsf{kdf}_{0RTT}(es, log_1) = k_c$ where
$\quad ets_c = \mathsf{derive\text{-}secret}(es, \mathsf{ets}_c, log_1)$
$\quad k_c = \mathsf{hkdf\text{-}expand\text{-}label}(ets_c, \mathsf{key}, \text{“”})$

Outline

Aperçu


Règlements européens
Directives eIDAS
ANSSI
RGS
CC, CSPN, Qualification
IGC/A
X.509
AC
e-gouv
Prestataires de services de confiance
Produits de sécurité
Industriels
NIST
FIPS 140
Services de confiance
OpenSS
AC
PKIX
Certificats EV
HTTPS
DNSSec
PGP
Prestataires
Signatura
LDAPS
Réseau de l'employeur
TLS
IKE
IPSec
3D-Secure
SET
Commerçant en ligne
TOR
Monkeysphere
Bitcoin
WWW
S/MIME
GPG
Internet
PGP
EMV
Banque
Commerçant

PKI : Public Key Infrastructure

- Utiliser des clefs publiques
- Établir une clef symétrique de session
- Confiance
- Certificats
- Authorité de certifications
- Chaîne de confiance

PKI

Problem: how to agree securely on a symmetric key?

- Face-to-face key exchange O(n^2) keys
- Key exchange via a trusted third party (TTP) Kerberos 5

Idea : Public-key encryption solves the problem of key exchange.
How to ensure the authenticity of other people's public keys?

- Face-to-face key exchange O(n)
- Key exchange via a trusted third party (TTP)

Other solution is : Key certificates.

Comment échanger une clef secrète en toute sécurité

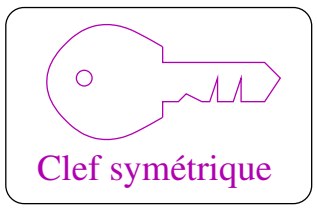
Plusieurs solutions :

- Protocole de Diffie-Hellman (Attaque Man-In-the-Middle)
- Kerberos utilise un tiers de confiance et des clefs symétriques
- Architectures à clefs publiques (PKI)

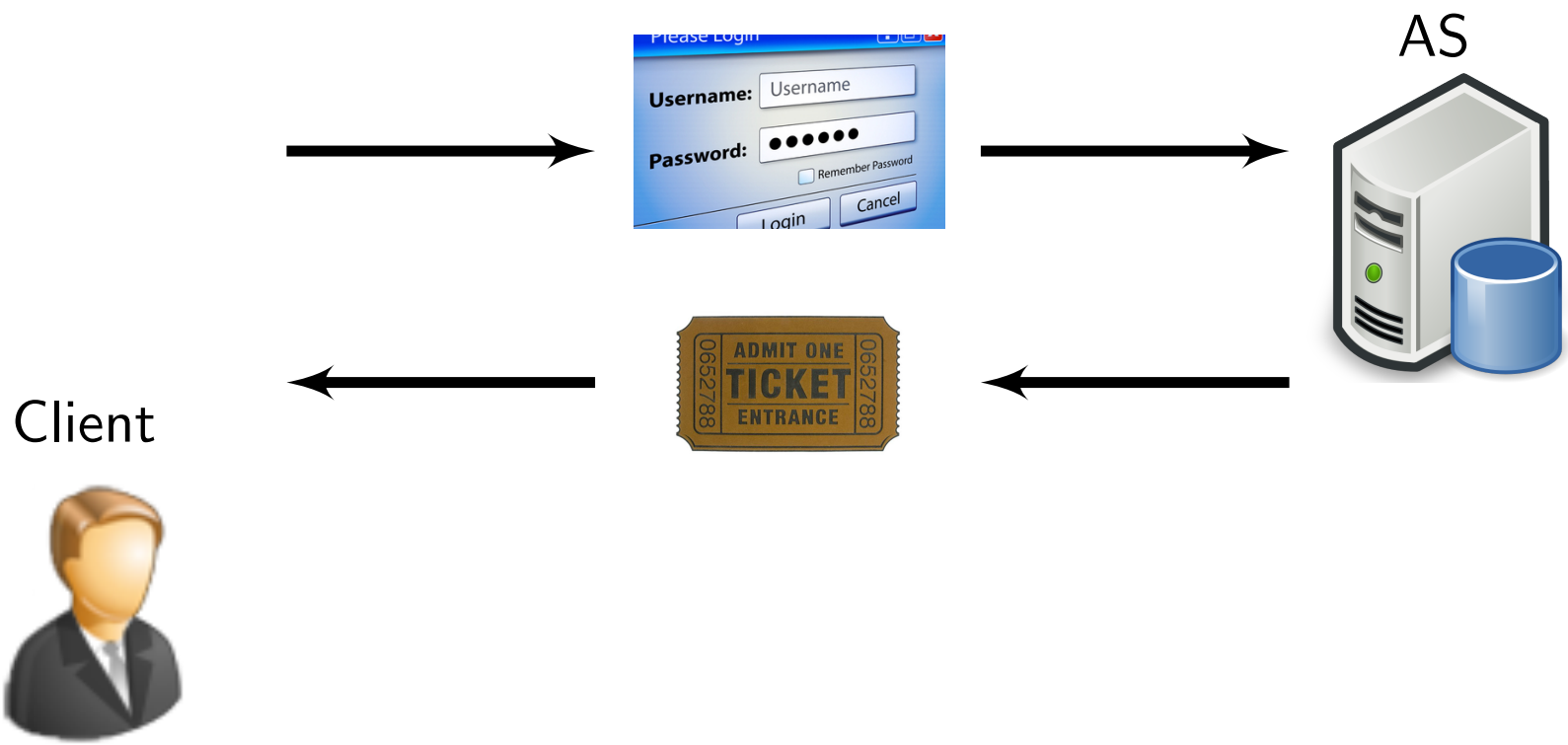
Kerberos V5

Utilise pour les communications:

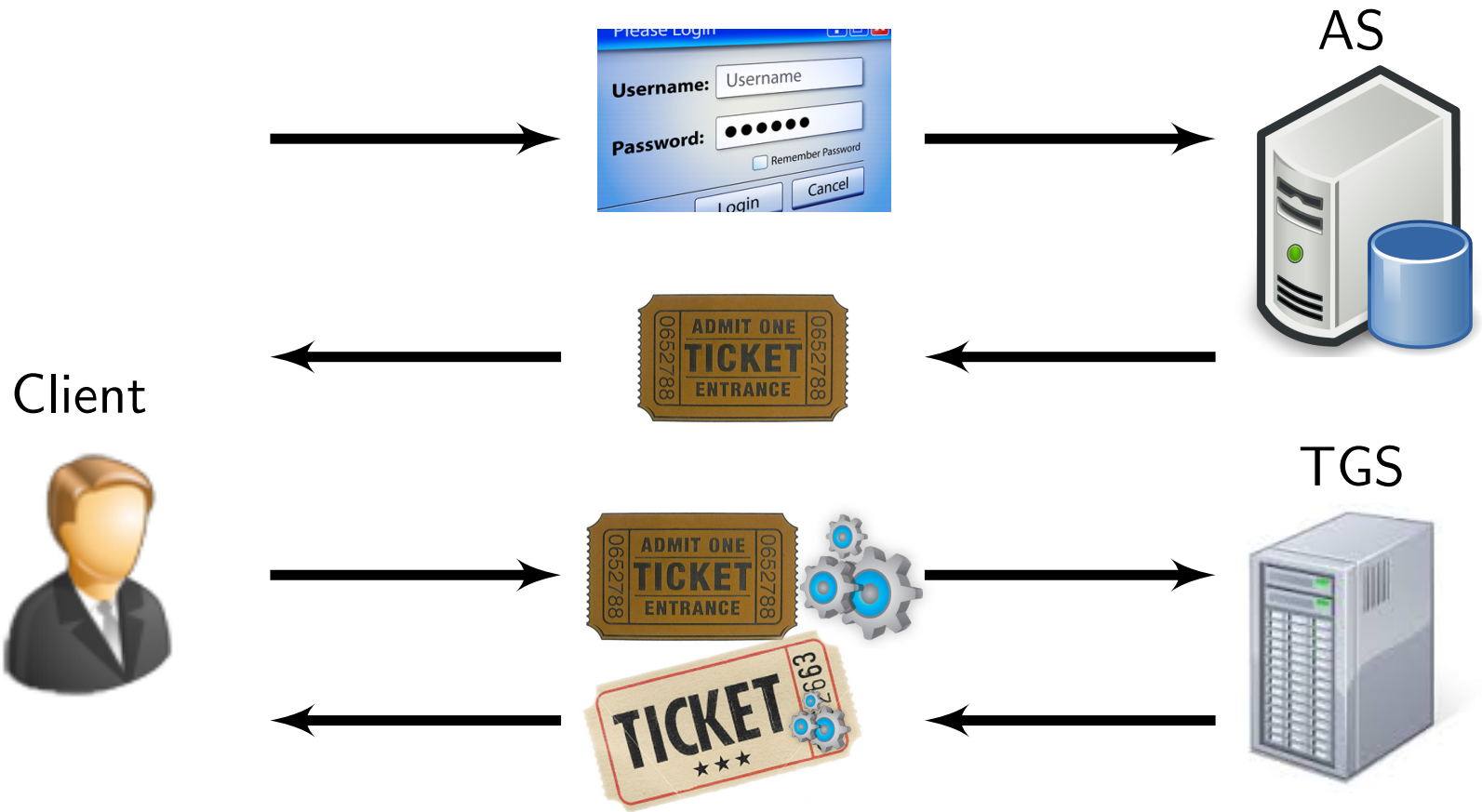
- un tiers de confiance : AS (Authentication Server)
- chiffrement symétrique (clefs privées)
- des tickets : TGS (Ticket Granting Service)
- des mots de passe



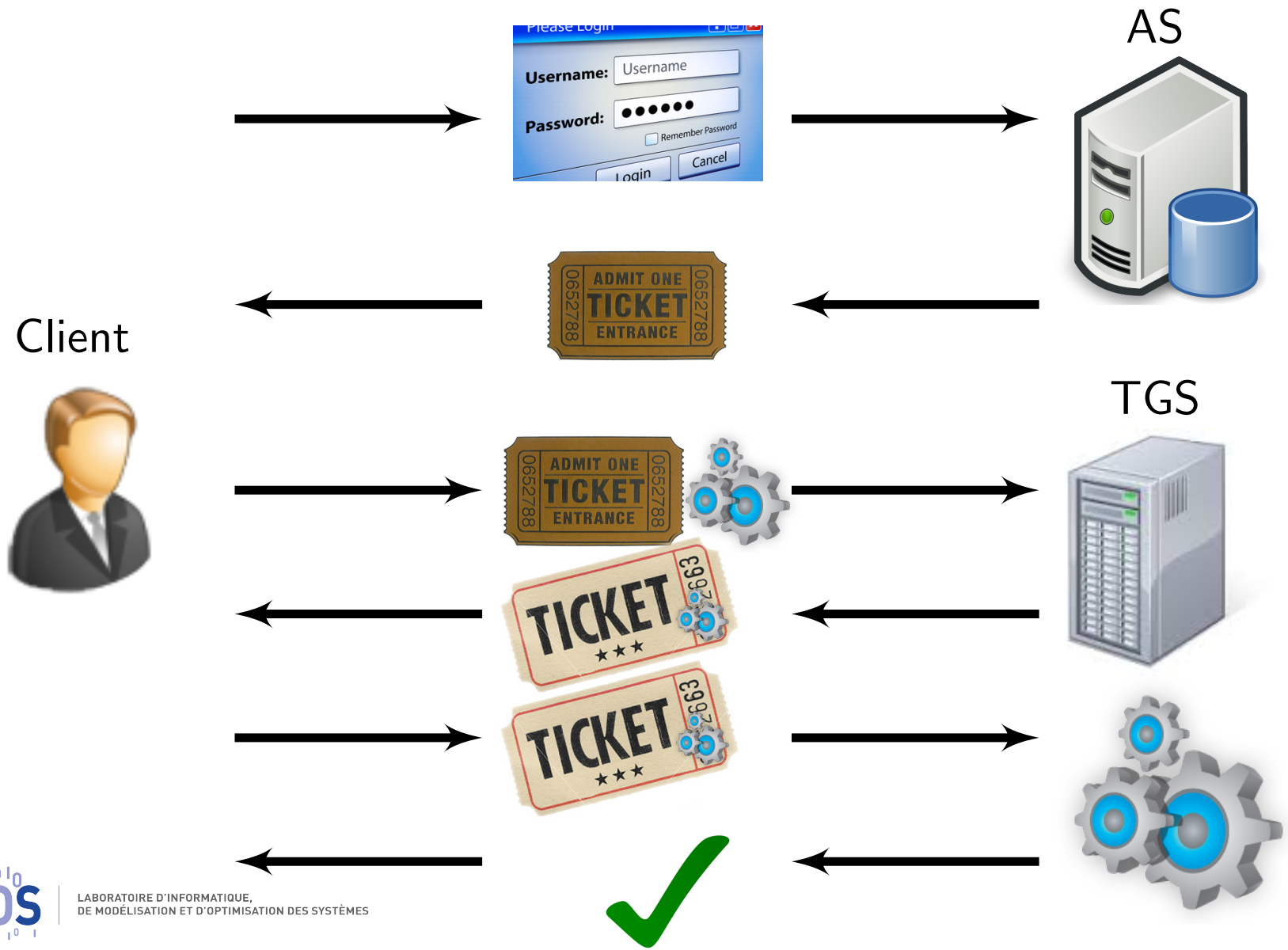
Kerberos V5: Principe en 3 phases



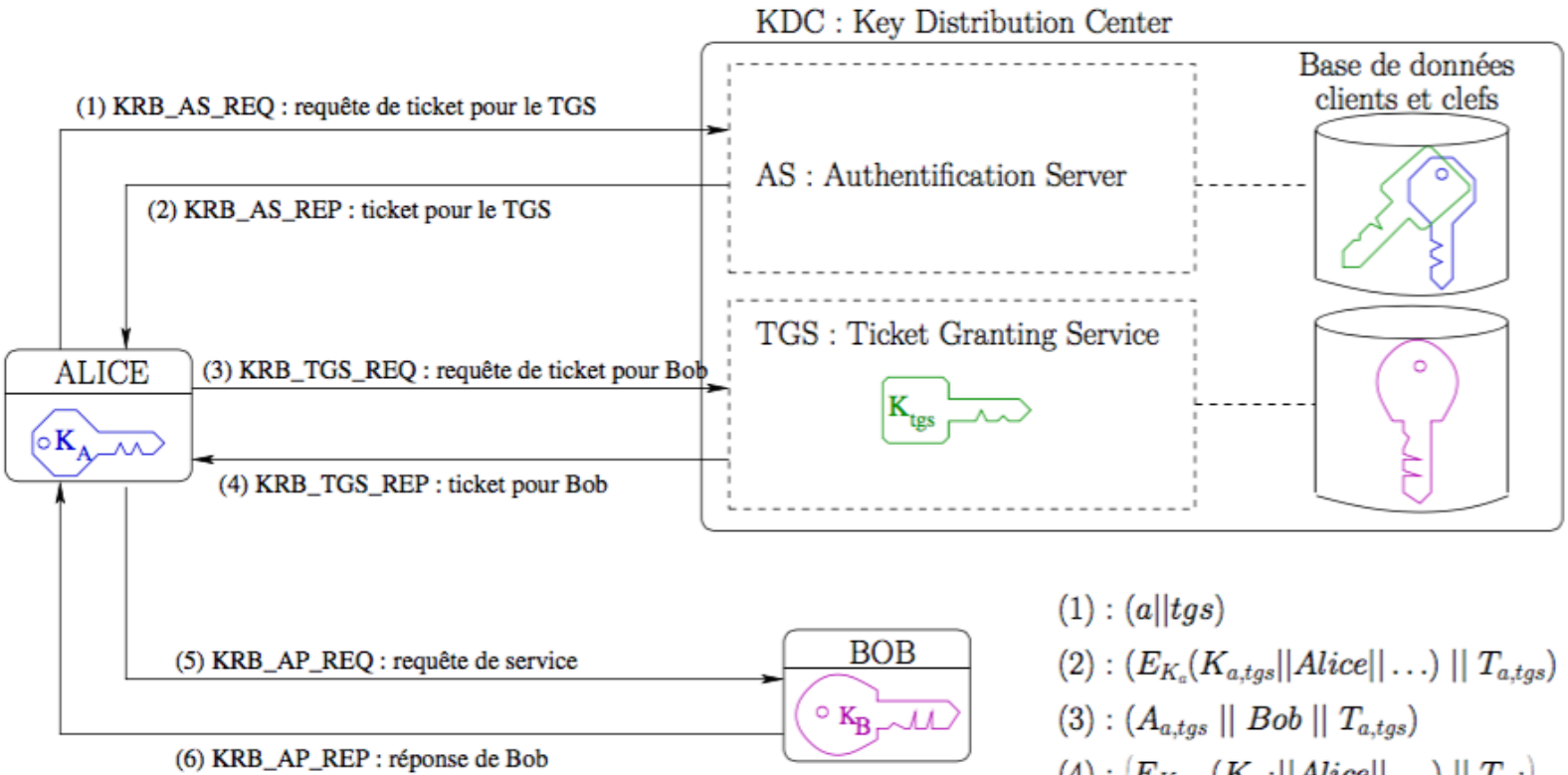
Kerberos V5: Principe en 3 phases



Kerberos V5: Principe en 3 phases



Kerberos V5



(1) : $(a||tgs)$

(2) : $(E_{K_a}(K_{a,tgs}||Alice||\ldots) \,||\, T_{a,tgs})$

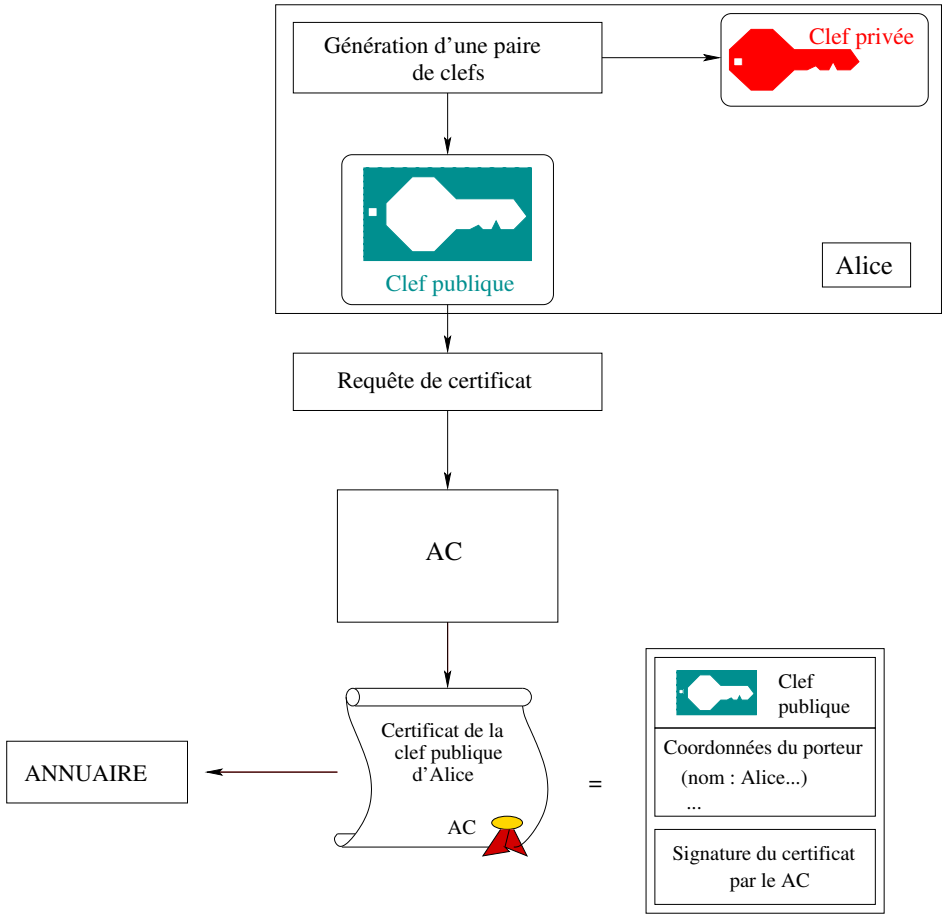
(3) : $(A_{a,tgs} \,||\, Bob \,||\, T_{a,tgs})$

(4) : $\left(E_{K_{a,tgs}}(K_{a,b}||Alice||\ldots) \,||\, T_{a,b}\right)$

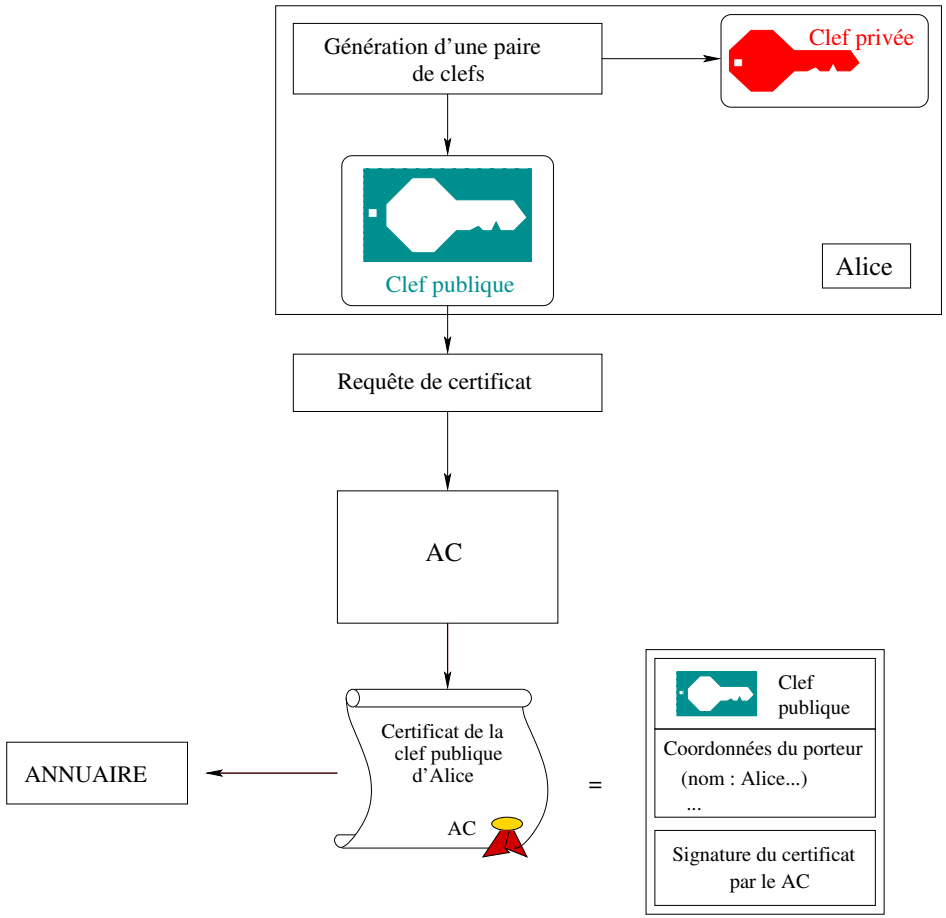
(5) : $(A_{a,b} \,||\, T_{a,b})$

(6) : $\left(E_{K_{a,b}}(t+1)\right)$

Public Key Infrastructure (PKI)



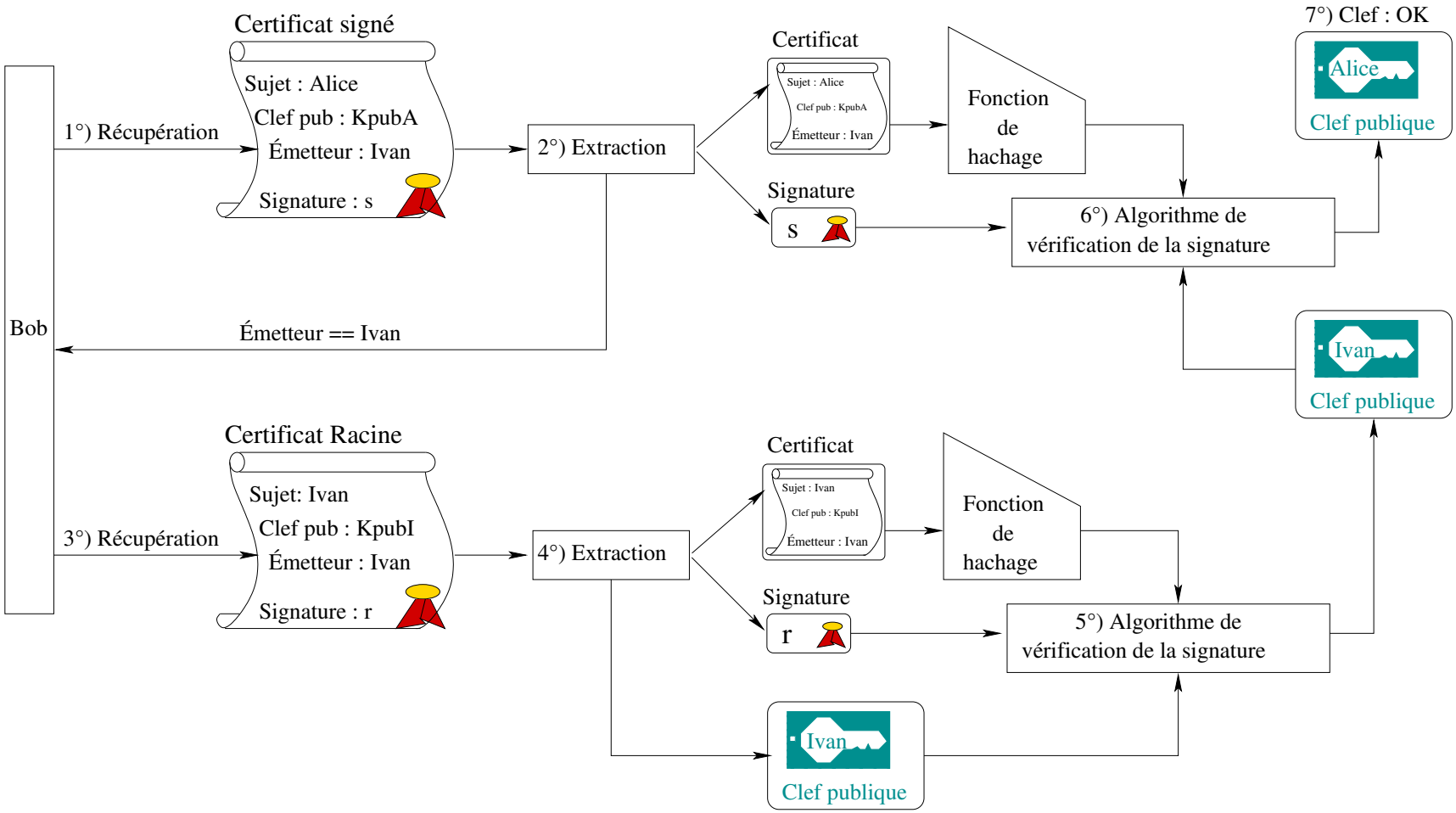
Public Key Infrastructure (PKI)



Authentification de l'AC confiance assurée par

- Chaîne de certificats
- Certificat racine ou certificat auto-signé

Vérification

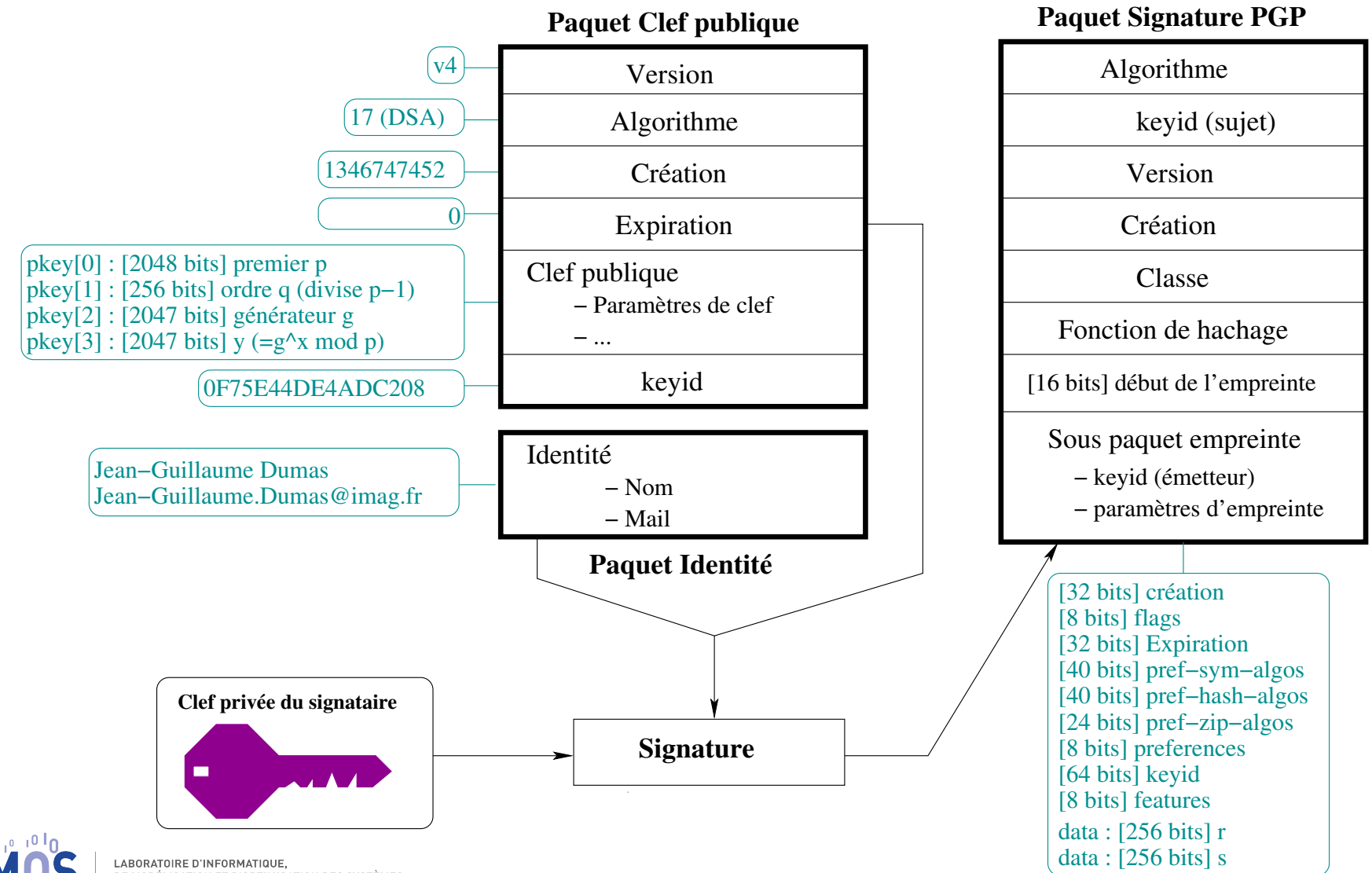


Différentes PKI

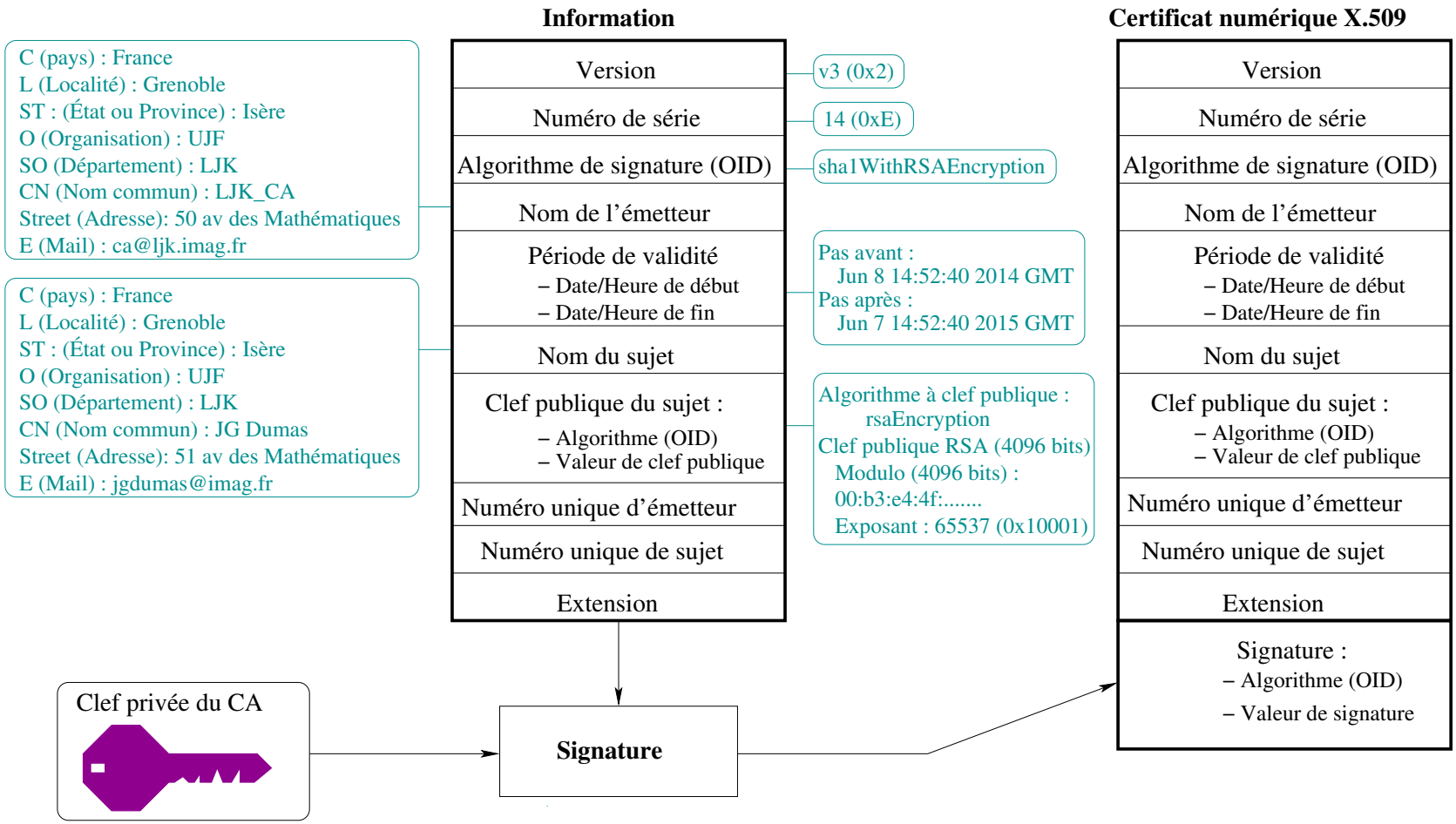
Différents modèles de confiance

- Hiérachique et ancre de confiance PKIX (PKI for X.509)
- Hiérarchique maillé et confiance distribuée
- Embarquée et magasins d'encre de confiance
- Non hiérarchique centré sur l'utilisateur (PGP)
- Autres : Simple PKI, Simple Distributed Security Infrastructure

Certificat PGP



Certificat X509



The concept of key certficate

Main idea

- Alice trusts Bob and knows his public key
- Bob has signed asserting that Carol's key is K
- Then Alice may be willing to believe that Carol's key is K.

Definition

A key certificate is an assertion that a certain key belongs to a certain entity, which is digitally signed by an entity (usually a different one).

Two Kinds of PKI

Hierarchical PKI

- Certificate Authoirities are different of users
- X.509 (PKIX)

Non-Hierarchical PKI

- Each user manages his own trust network
- Pretty Good privacy (PGP) and P2P based PKI

Others : SDSI, SPKI

Example “https” for gmail

- Gmail sends to your browser its public key and a certificate signed by a certificate authority ”Thawte Consulting (Pty) Ltd.” to prove that this key really is gmail’s key.
- Your browser will verify Thawte’s signature on gmail’s key using the public key of this reputable key certificate authority, stored in your browser.
- Hence your browseer trust Gmail.

Trust chains

Example

A1 has signed asserting that A2's key is K2
A2 has signed asserting that A3's key is K3
. . .
A18 has signed asserting that A19's key is K19
A19 has signed asserting that A20's key is K20
A20 has signed asserting that B's key is K

- If I know A1's key, and I trust A1, A2,..., A20, then I am willing to believe that B's key is K.
- A1 states that A2's key is K2. So, if A2 signs an assertion X, then A1 states that A2 states X. Thus, in the situation above, A1 states that A2 states that A3 states ... that A19 states that A20 states that B's key is K.

PKI functionalities

- Creation of keys
- Authentication of public keys
- Diffusions of certificates
- Veification of ceritficates
- Revocation of certificates
- Others :
  - Protection of private key
  - Journalisation of actions
  - Revocations of privates keys
  - Storage of certifcates

Definition PKI

PKI is an infrastructure build of certificates and servers to create, manage and publish certificate to allow autenticity certified by an authority.

X.509 certificates used by SSL/TLS, SET, S/MIME, IPSec ...

X.509 components:

- Version
- Serial number
- Signature algorithm identifier
- Issuer name
- Period of validity
- Subject name
- Subject public-key and algorithm
- Issuer unique identifier
- Subject unique identifier
- Extensions
- Signature

X.509 certificates are typically issued by a certificate authority (CA). Anyone can be a CA, but not all CA's are trusted by everyone.

Pretty Good Privacy (PGP) by Phil Zimmermann, 1991

Generate keys for you, and help you manage them

A "PGP key" has several parts:

- the name of its owner
- the numerical value(s) comprising the key
- what the key is to be used for
- the algorithm the key is to be used with
- (possibly) an expiration date

Software: OpenPGP, or GnuPG

PGP

PGP stores lots of different keys for

- signing keys or emails or ...
- encrypting
- your own secret key (this will be stored encrypted with a passphrase)
- your own public key and the public keys of your friends and associates (stored in the clear)

The PGP software puts them in a file, called your keyring.

PGP

- Your private keys are in a file only you can read; for extra security, they are stored encrypted with a pass phrase.
- The public keys don't have to be protected.
- The keyring also stores certificates, i.e. copies of other people's public keys which are signed by you. These ones are known with certainty by you to belong to the people they claim to belong to.

PGP: How to send a message to someone

A "signed message"

PGP signs a hash of the message.

A message encrypted with their public key

- PGP encrypts it with a newly-generated symmetric key
- You send that encrypted version appended to the symmetric key encrypted with the public key.

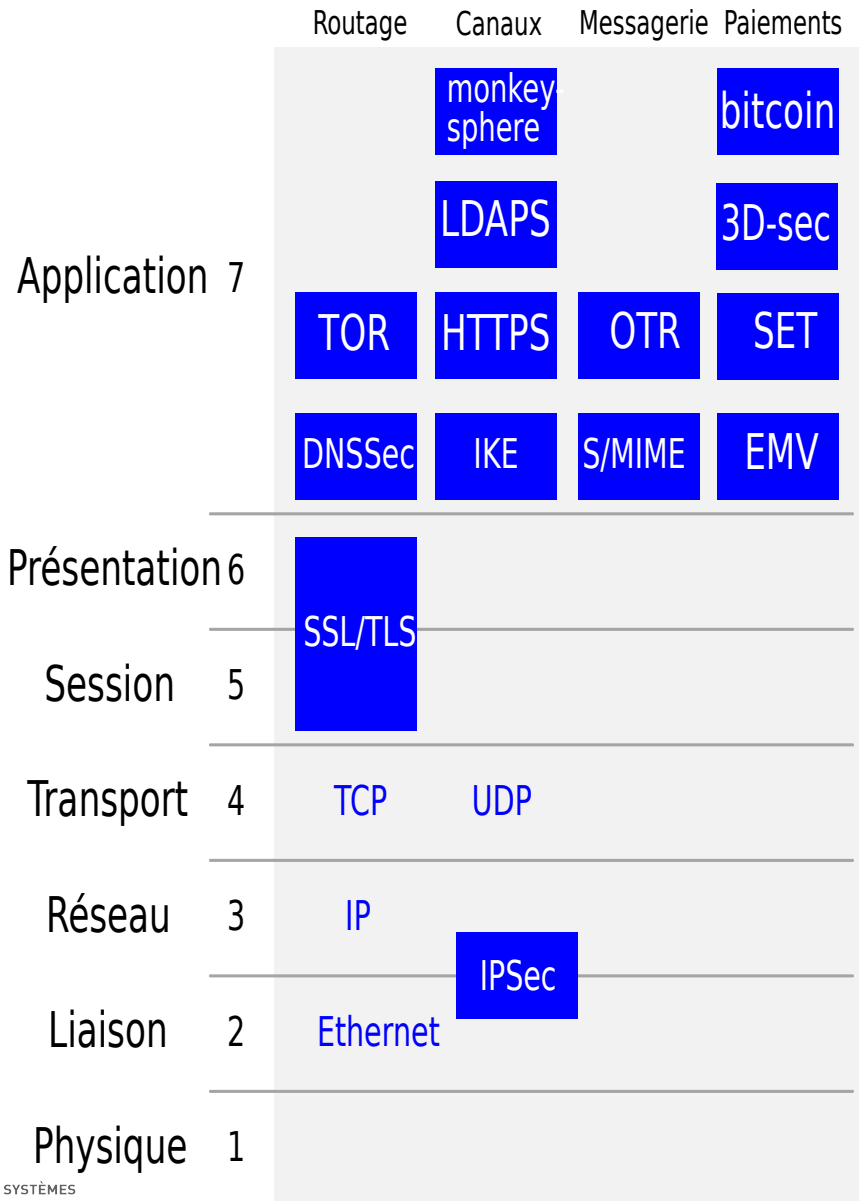
Why does no-one use PGP?

- It's not considered necessary.
- It's quite complicated. You need to spend a day to understand it properly. And even then, understanding is not guaranteed!
- It's a hassle. You need to maintain your keys, your web of trust, you need to configure your mail client.

Why Johnny can't encrypt is an article explaining why people can't/don't want to use PGP.
USER CONFIDENCE is among the most difficult ones in computer security.

Outline

Applications



Application : HTTPS

HTTP Secure utilise SSL/TLS sur le port 443 et assure

- Confidentialité
- Intégrité

Le serveur Web est authentifié par un certificat X.509.

Application : HTTPS

HTTP Secure utilise SSL/TLS sur le port 443 et assure

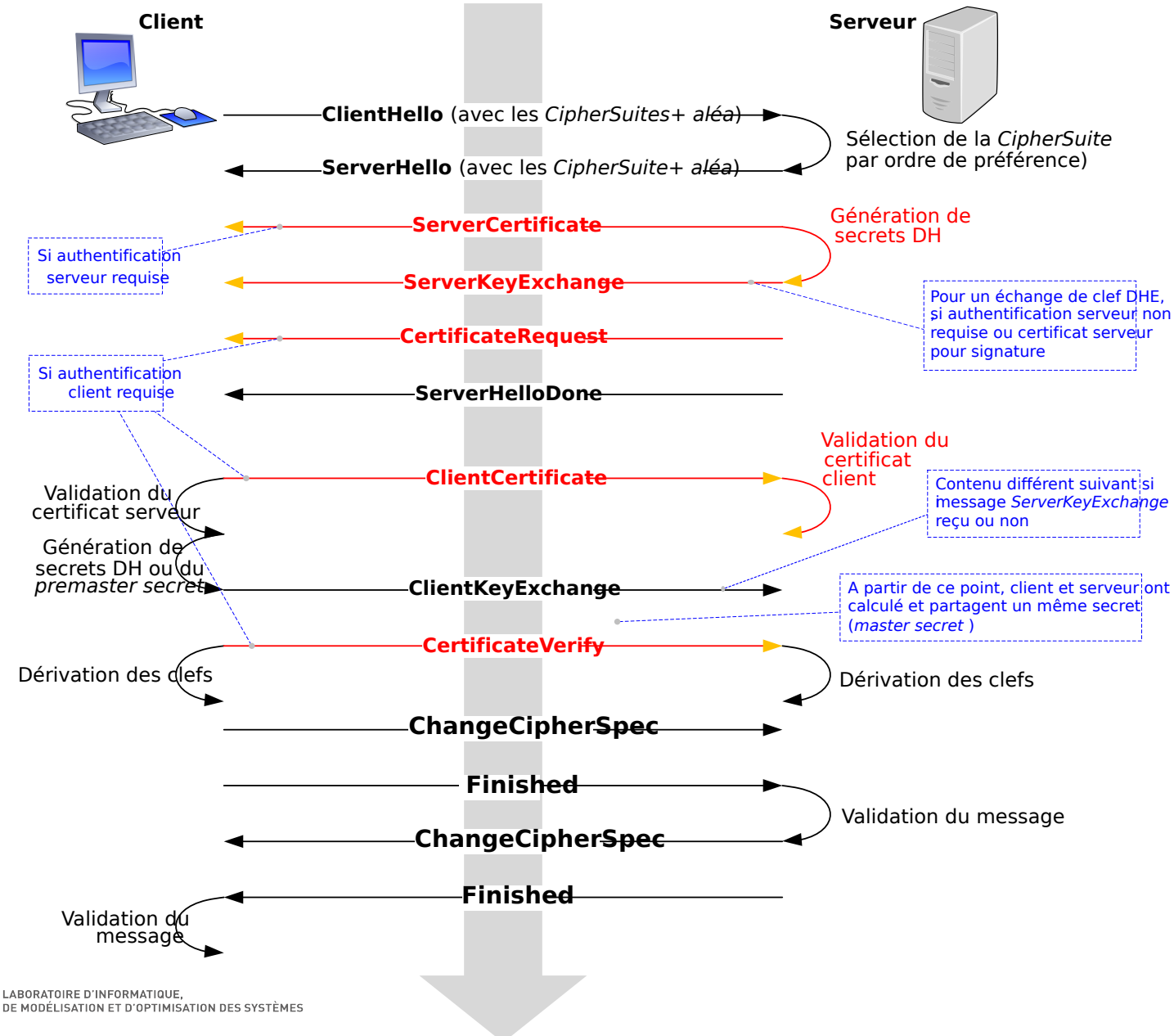
- Confidentialité
- Intégrité

Le serveur Web est authentifié par un certificat X.509.

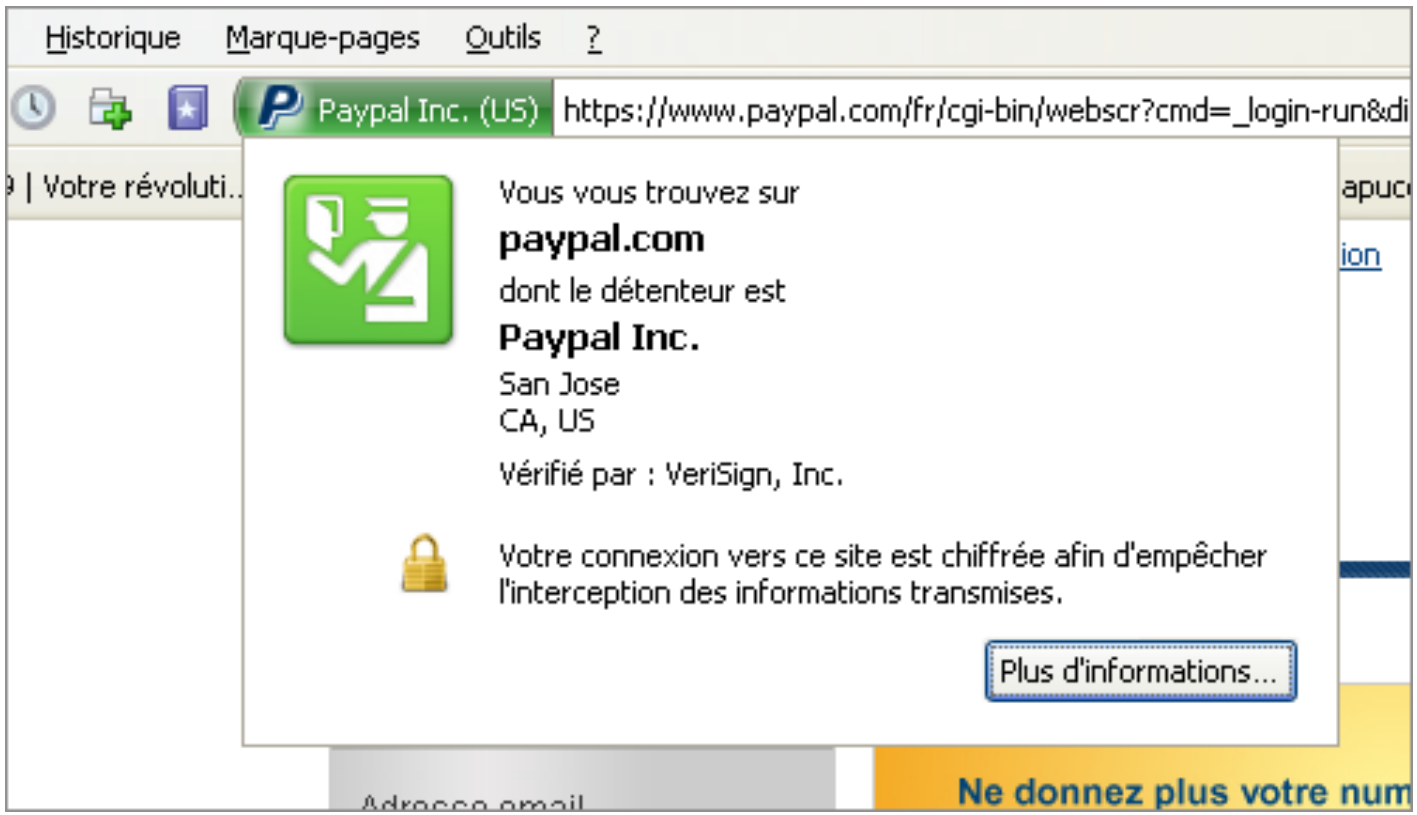
TLS = 5 protocoles

- *Handshake*, connexion sécurisée entre le client et le serveur
- *Change Cipher Spec*, nouvelle clef de session va être utilisée
- *Alert* $\Rightarrow$ Warning ou Fatal
- *Application Data*, encapsulation des données après Handshake
- *TLS Record*, encapsule puis relaye les données

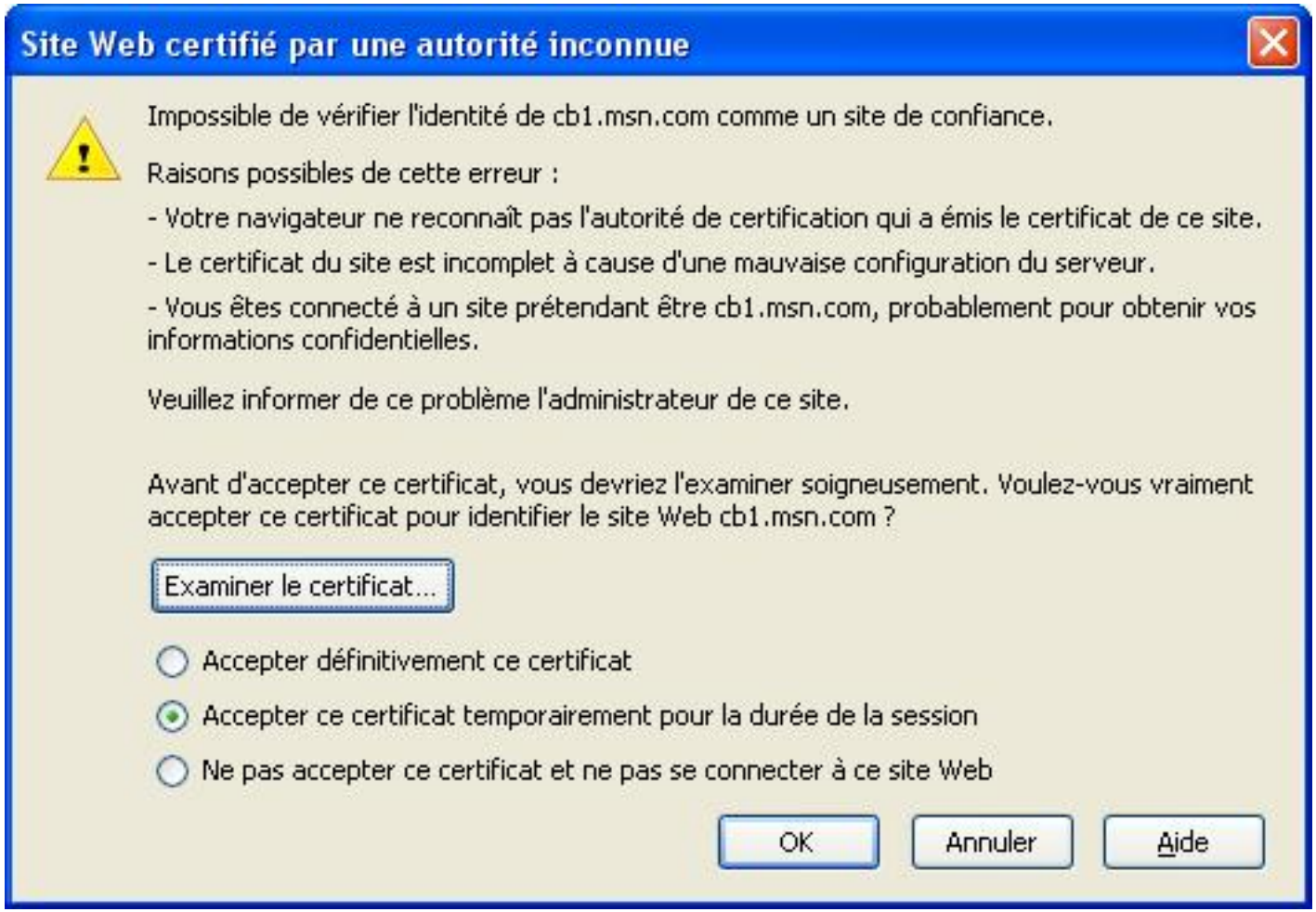
Application : TLS Handshake



Application :



Application :



AC est inconnue du magasin de certificats.

Application :

Cette connexion n'est pas certifiée

Vous avez demandé à Iceweasel de se connecter de manière sécurisée à **static.ak.facebook.com**, mais nous ne pouvons pas confirmer que votre connexion est sécurisée.

Normalement, lorsque vous essayez de vous connecter de manière sécurisée, les sites présentent une identification certifiée pour prouver que vous vous trouvez à la bonne adresse. Cependant, l'identité de ce site ne peut pas être vérifiée.

Que dois-je faire ?

Si vous vous connectez habituellement à ce site sans problème, cette erreur peut signifier que quelqu'un essaie d'usurper l'identité de ce site et vous ne devriez pas continuer.

Sortir d'ici !

Détails techniques

static.ak.facebook.com utilise un certificat de sécurité invalide.

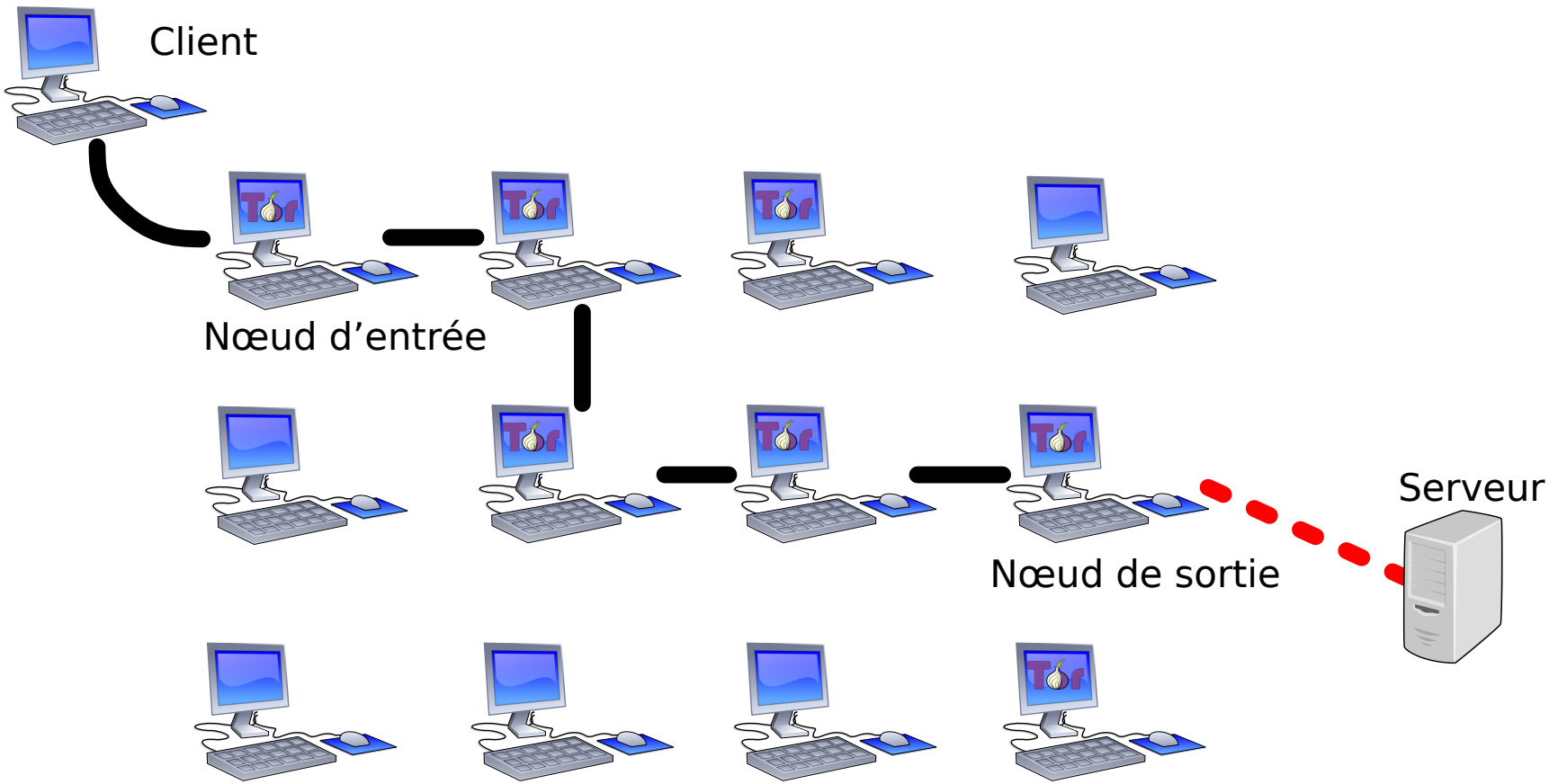
Le certificat n'est valide que pour les noms suivants :
a248.e.akamai.net , *.akamaihd.net , *.akamaihd-staging.net

(Code d'erreur : ssl_error_bad_cert_domain)

Je comprends les risques

- Soit le site Web est faux
- Soit des certificats distincts sont créés pour des sites distincts
- Soit il faut ajouter une valeur dans un champ

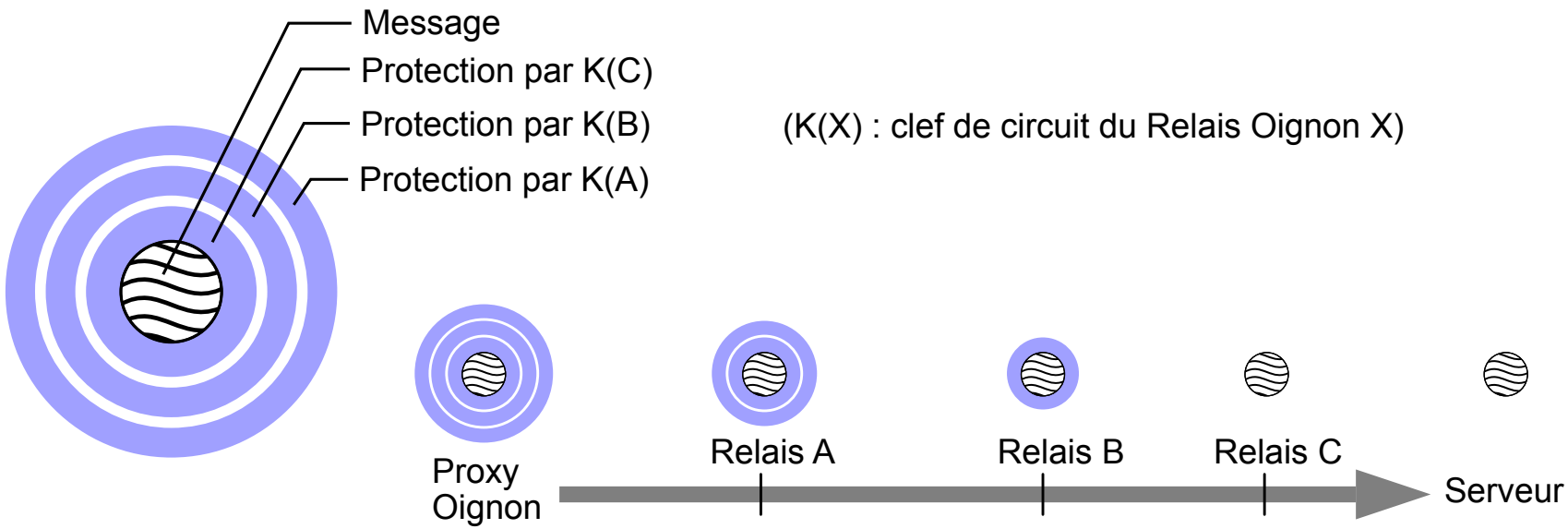
Application : The Onion Router



https://www.torproject.org

LIMOS LABORATOIRE D'INFORMATIQUE, DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

Application : Tor



Application : Messagerie instantanée

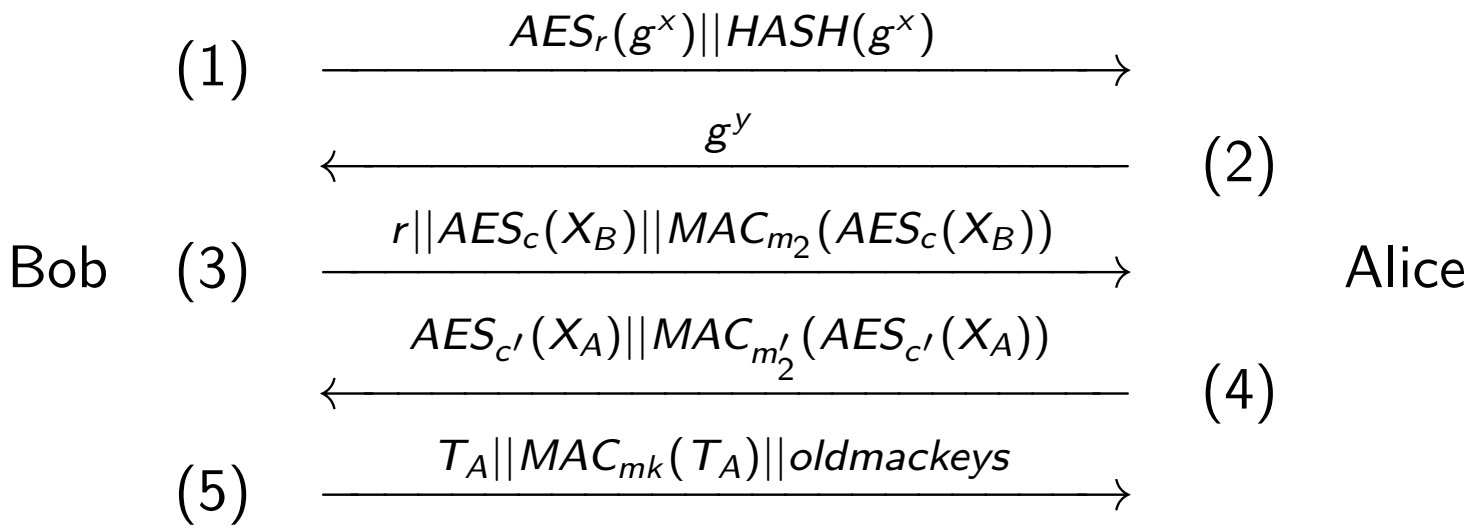
Application : Off-the-Record Messaging (OTR)

Inventé par N. Borisov, I. Goldberg et E. Brewer en 2004.

- Confidentialité : Personne ne peut lire vos messages
- Authentification : Sûr de parler à son interlocuteur
- Révocabilité (*deniability*) des conversations : personne ne doit pouvoir prouver que vous êtes l'auteur des messages.
- Les messages sont authentiques et non-modifiés
- Confidentialité persistante (Perfect forward secrecy) : La perte des clefs privées ne compromet pas les conversations passées.

Utilise AES, SHA-1, Diffie-Hellman dans le protocole AKE

Application : AKE



À partir de $s := (g_y)^x$ génére par hachage :

- 2 clefs symétriques c et c'
- 4 clefs MAC m_1, m'_1, m_2 et m'_2

$X_B := Kpub_B||keyid_B||SIGB_(M_B)$
$X_A := Kpub_A||keyid_A||SIG_A(M_A)$
$M_B := MAC_{m_1}(g^x||g^y||Kpub_B||keyid_B)$;
$M_A := MAC_{m'_1}(g^y||g^x||Kpub_A||keyid_A)$;
$T_A := (keyid_A||keyid_B||next_{dh}||ctr||AES - CTR_{ek,ctr}(msg))$

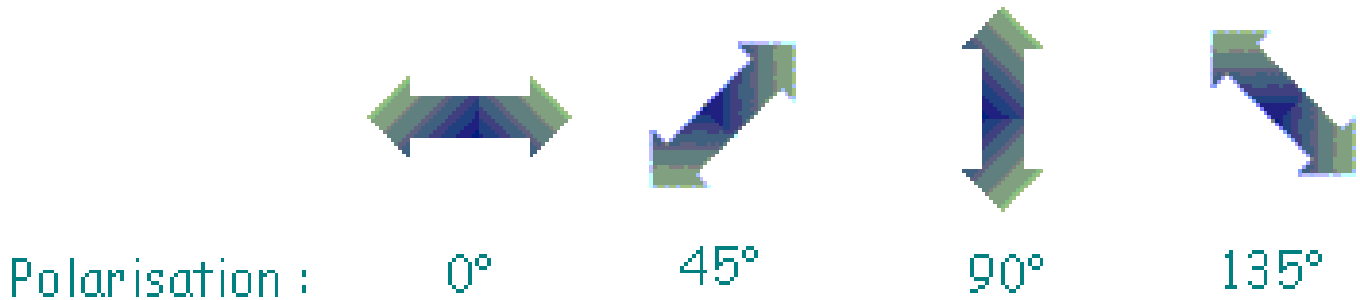
Outline

Quantum Channel

- A quantum communication channel which allows quantum states to be transmitted: In the case of photons this channel is generally either an optical fibre or simply free space.
- In addition they communicate via a public classical channel, for example using radio waves or the internet.
- Neither of these channels need to be secure; the protocol is designed with the assumption that an eavesdropper (referred to as Eve) can interfere in any way with both.

Quantum

Using photon polarization states to transmit the information.



- No cloning theorem
- Reading a photon will change his state.

BB84 is a quantum key distribution scheme developed by Charles Bennett and Gilles Brassard in 1984.

BB84 (I)

Alice sends photons with one of the four polarizations, which she chooses at random.

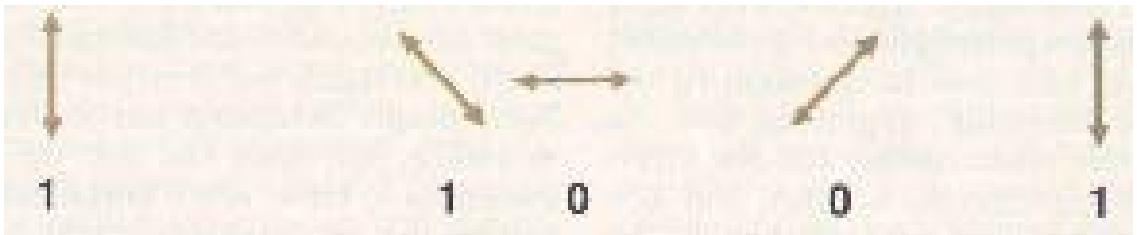
For each photon, Bob chooses at random the type of measurement: either the rectilinear type (+) or the diagonal type (X).

Bob records the result of his measurements but keeps it a secret.

BB84 (II)

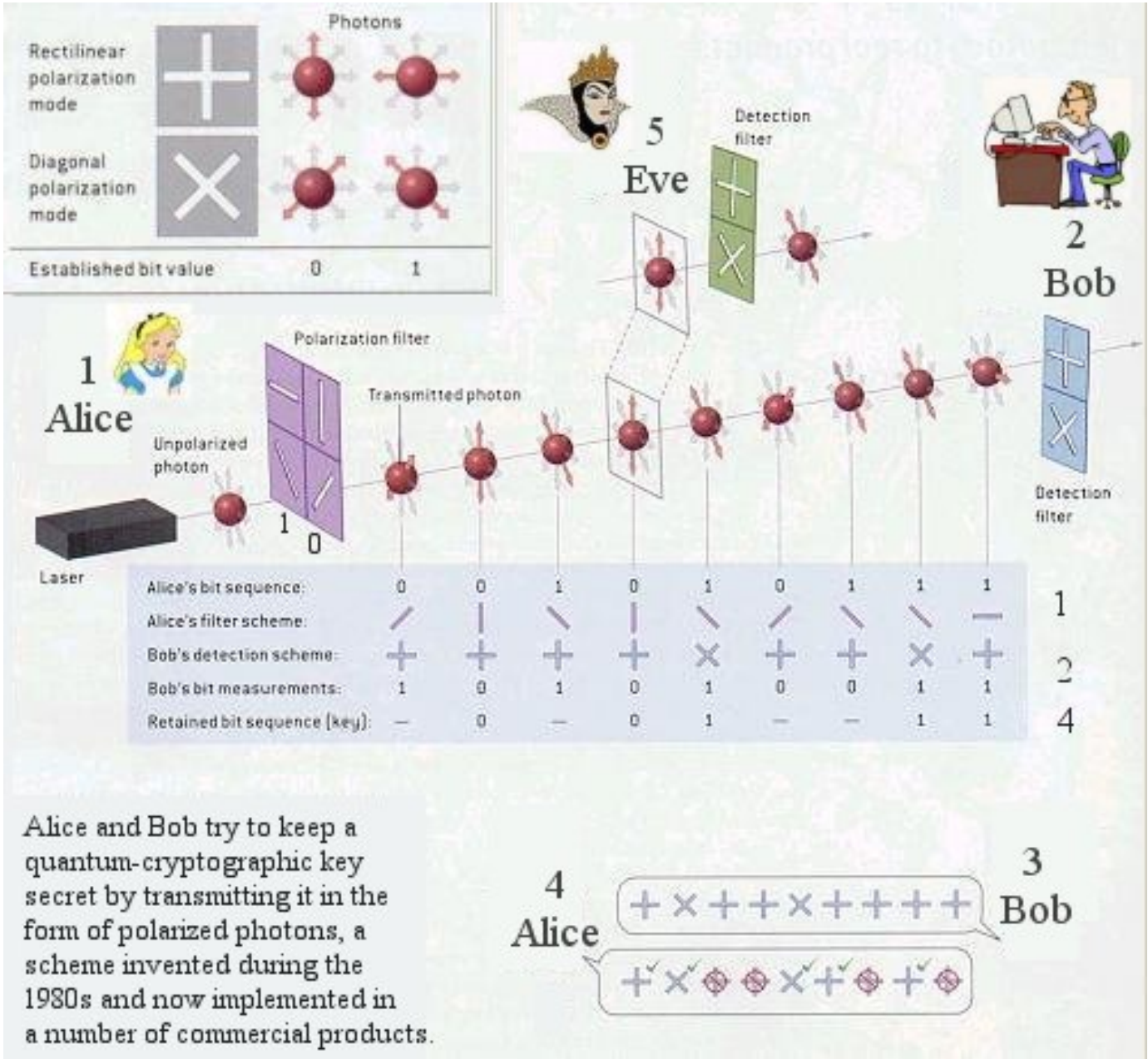
After the transmission, Bob tells Alice the measurement types he used (but not his results) and Alice tells him which were correct for the photons she sent. This exchange may be overheard.

Alice and Bob keep all cases in which Bob should have measured the correct polarization. These cases are then translated into bits (1s and 0s) to define the key.



Intruder Dectection

- Alice and Bob compare a subset of remaining bit strings.
- Eavesdropper should have introduce error on Bob side
- If more than p bits differ they abort the key and try again


Photons
Rectilinear polarization mode
Diagonal polarization mode
Established bit value 0 1
5
Eve
Detection filter
2
Bob
1
Alice
Polarization filter
Transmitted photon
Unpolarized photon
Laser
Detection filter
Alice's bit sequence: 0 0 1 0 1 0 1 1 1
Alice's filter scheme:
Bob's detection scheme:
Bob's bit measurements: 1 0 1 0 1 0 0 1 1
Retained bit sequence (key): – 0 – 0 1 – – 1 1
1
2
4
Alice and Bob try to keep a quantum-cryptographic key secret by transmitting it in the form of polarized photons, a scheme invented during the 1980s and now implemented in a number of commercial products.
4
Alice
3
Bob

Outline

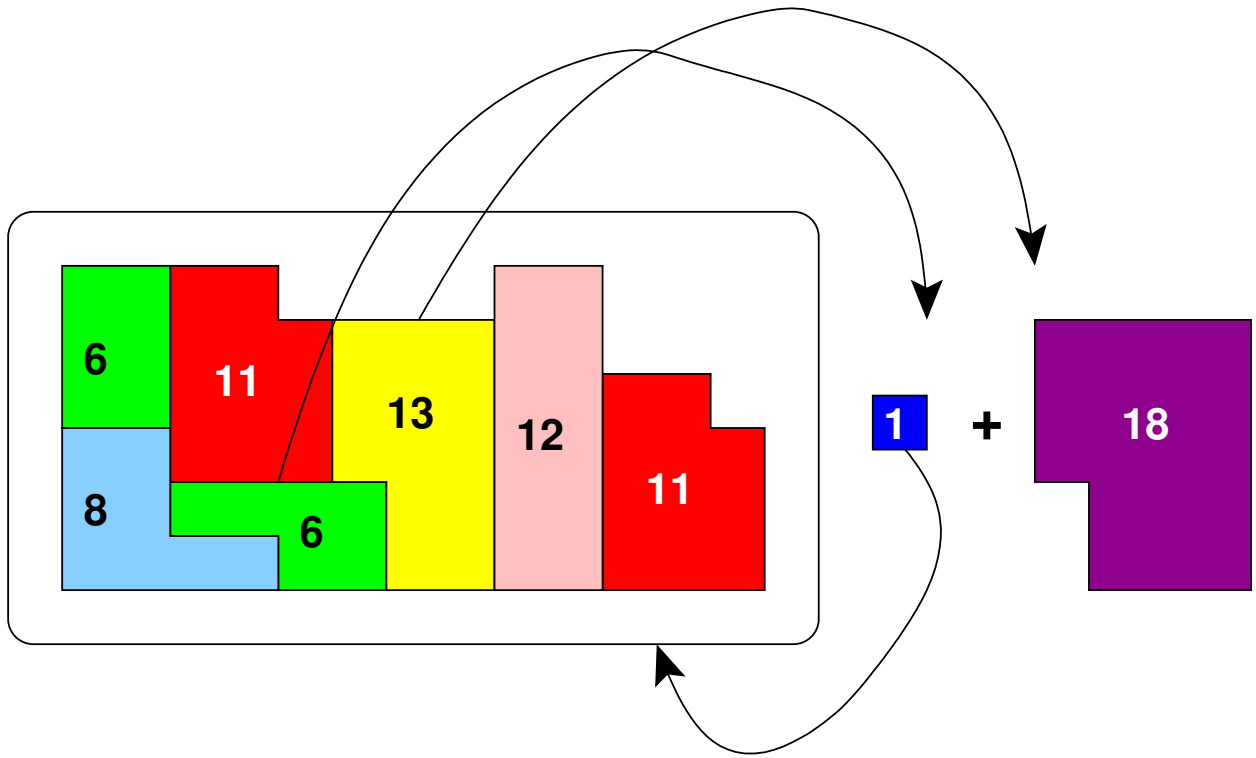
Bitcoin : monnaie électronique

Créée en 2008 par Satoshi Nakamoto (1 BTC $\approx$ 945 euros)

1	BTC = 1 Bitcoin	
0,01	BTC = 1 cBTC	= 1 centiBitcoin (ou bitcent)
0,001	BTC = 1 mBTC	= 1 milliBitcoin
0,000 001	BTC = 1 μBTC	= 1 microBitcoin
0,000 000 01	BTC = 1 Satoshi	

Taux de change du bitcoin

Payer 18 BTC avec des pièces



Clef symétrique

Exemples

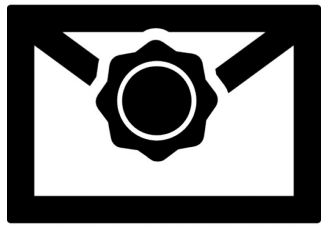
- DES
- AES

Chiffrement à clef publique

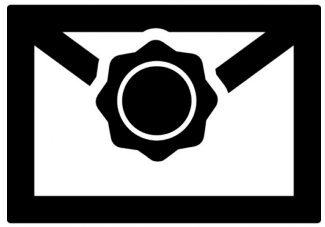
Exemples

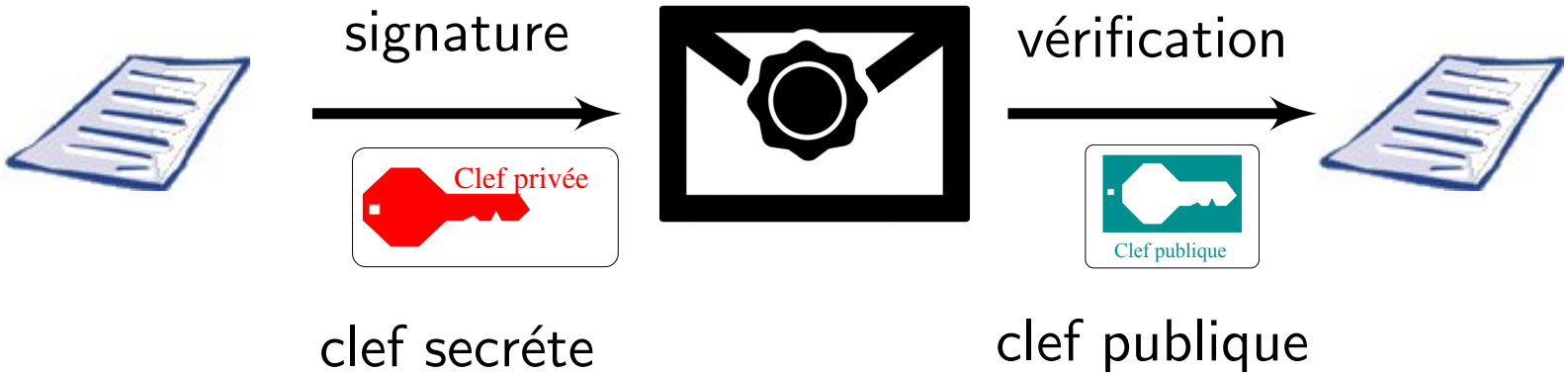
- RSA : $c = m^e \mod n$
- ElGamal : $c \equiv (g^r, h^r \cdot m)$

Signature



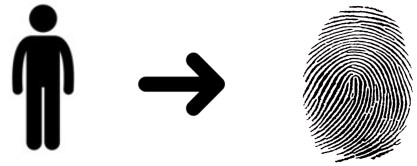
Signature



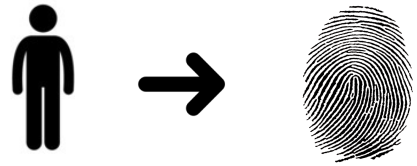


RSA: $m^d \mod n$

Fonction de Hachage (RIPEMD-160, SHA-256, SHA-3)

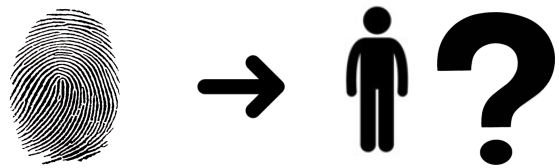


Fonction de Hachage (RIPEMD-160, SHA-256, SHA-3)

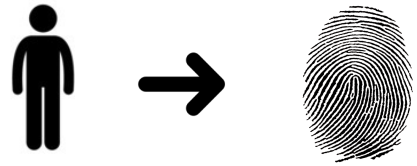


Propriétés de résitance

- Pré-image

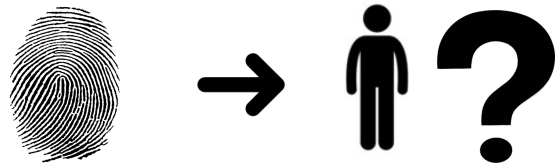


Fonction de Hachage (RIPEMD-160, SHA-256, SHA-3)

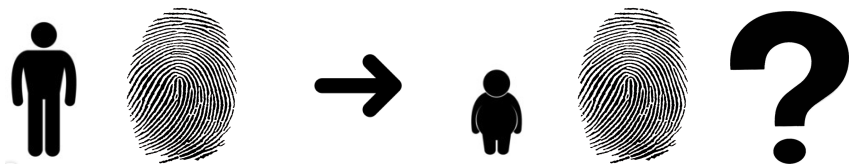


Propriétés de résitance

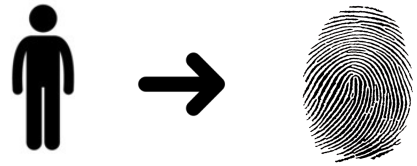
- Pré-image



- Seconde Pré-image

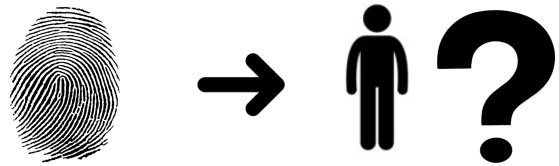


Fonction de Hachage (RIPEMD-160, SHA-256, SHA-3)

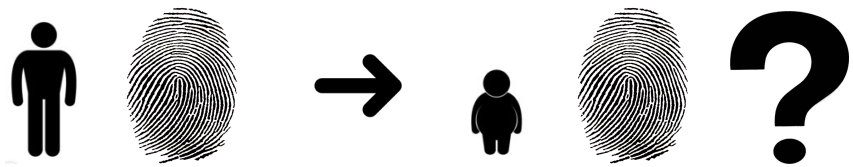


Propriétés de résitance

- Pré-image



- Seconde Pré-image



- Collision

Propriétés d'une monnaie électronique

- Non-Falsifiable (Unforgeable)

- Eviter la double dépense & identifiation fraudeur & "présemption d'innocence"

- Respect de la vie privée :
  - Anonymat faible : non identification d'un acheteur
  - Anonymat fort : non traçabilité d'un acheteur

Bitcoins : caractéristiques

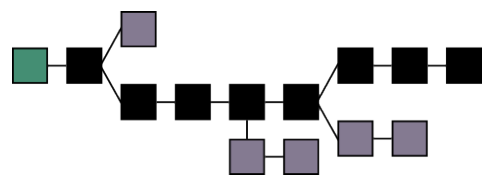
- Le nombre total de bitcoins est **fini**

  21 millions BTC

- Les transactions utilisent des **PKI**
- Numéro de compte :

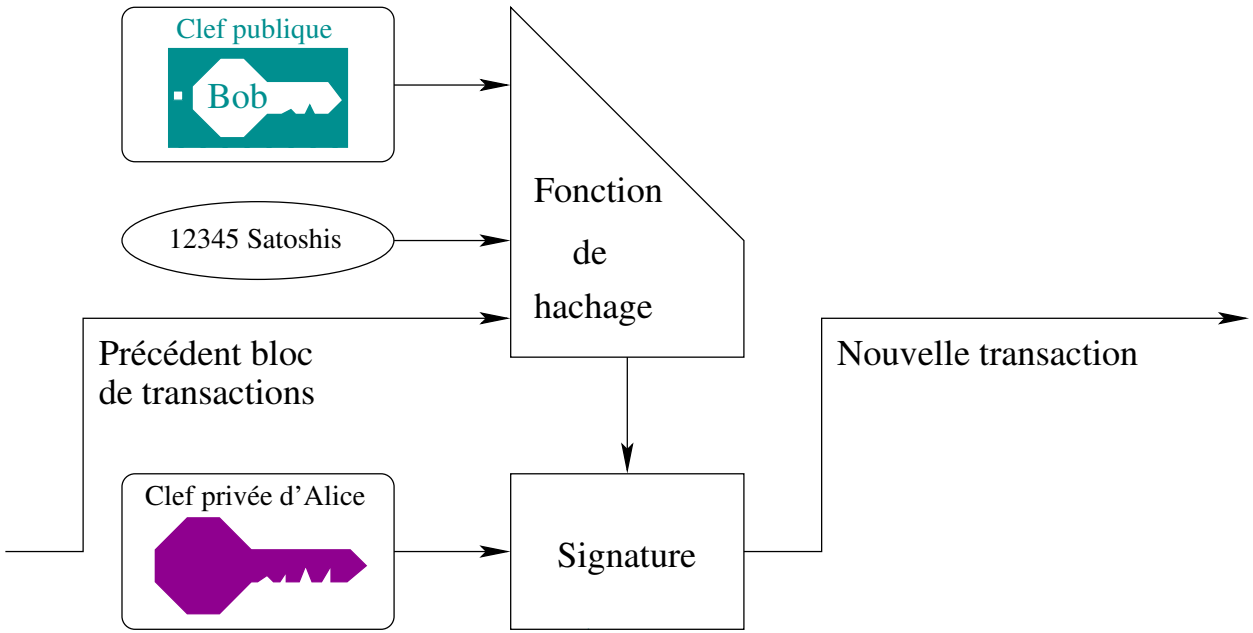
  $$\text{RIPEMD-160}(\text{SHA-256}(\text{ECDSA}_{pub}))$$

- Toutes les transactions sont **publiques**
- **Blockchain** : un système pair-à-pair qui garantit la validité des transactions

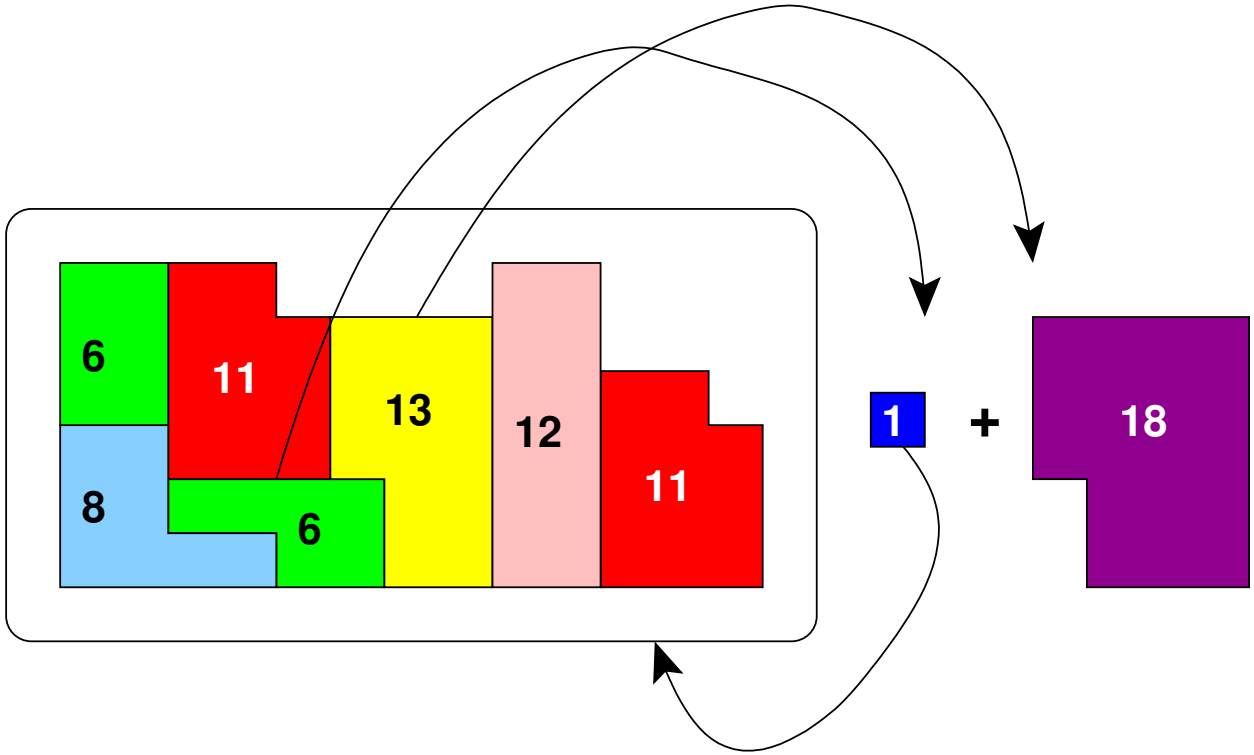


Comment faire une transaction?

Alice donne 12345 Satochis ($\approx 5c$) à Bob.



Payer 18 BTC avec des pièces

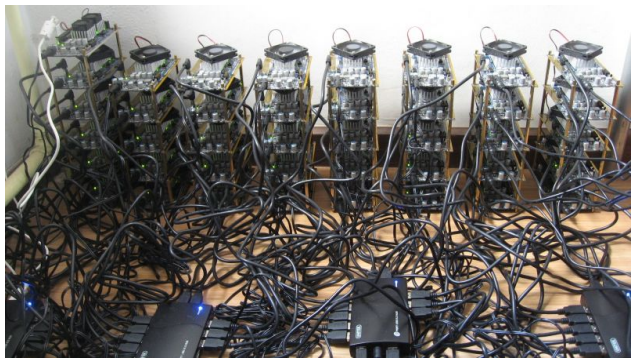


- Seuls des bitcoins possédés peuvent être dépensés

Miner des Bitcoins

Miner des Bitcoins

Les “*mineurs*” valident les transactions contre des bitcoins



Miner des Bitcoins

- Valider = résoudre un **objectif de hachage**
- Récompense initiale 50 BTC pour une validation
- Divisée par 2 tous les 210000 validations

$$\sum_{i=0}^{32} \frac{50}{2^i} \times 210\,000 = 21 \text{ millions BTC}$$

Miner : Objectif de hachage

Cible = 0000000000000000254845fa930deac4086b3e3bce21147e93f463b206d8076

Trouver une nombre n tel que

$$\text{SHA-256}(\text{SHA-256}(\text{Transactions}, n)) = x < \textit{Cible}$$

Avoir un 0 plus de au début de x

Miner : Objectif de hachage

Cible = 0000000000000000254845fa930deac4086b3e3bce21147e93f463b206d8076

Trouver une nombre n tel que

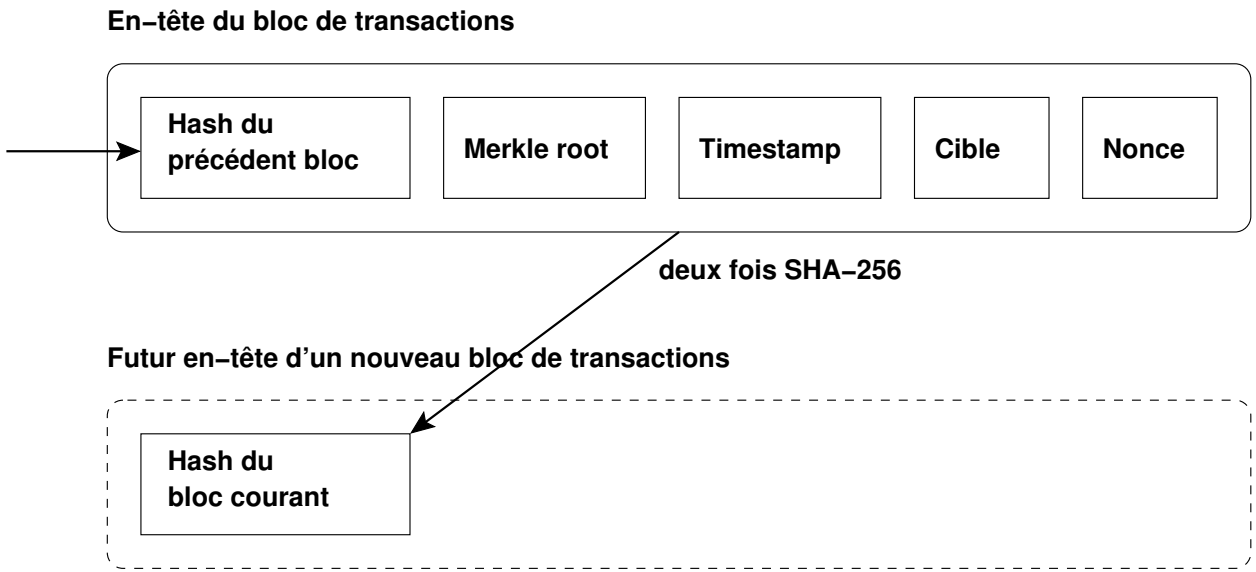
$$\text{SHA-256}(\text{SHA-256}(\text{Transactions}, n)) = x < \textit{Cible}$$

Avoir un 0 plus de au début de x

Stratégie : brute force

Tester toutes les valeurs possibles de n

Miner : Proof of work



Avoir un zéro de plus au début

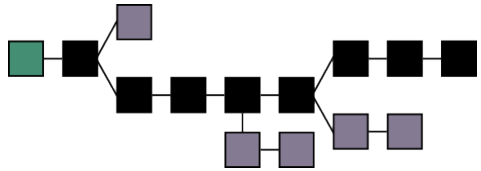
SHA-256(SHA-256(en-tête de bloc))

- les transactions passées (95 Go)
- les transactions à valider
- les secondes depuis 01/01/1970
- un nonce
- etc ...

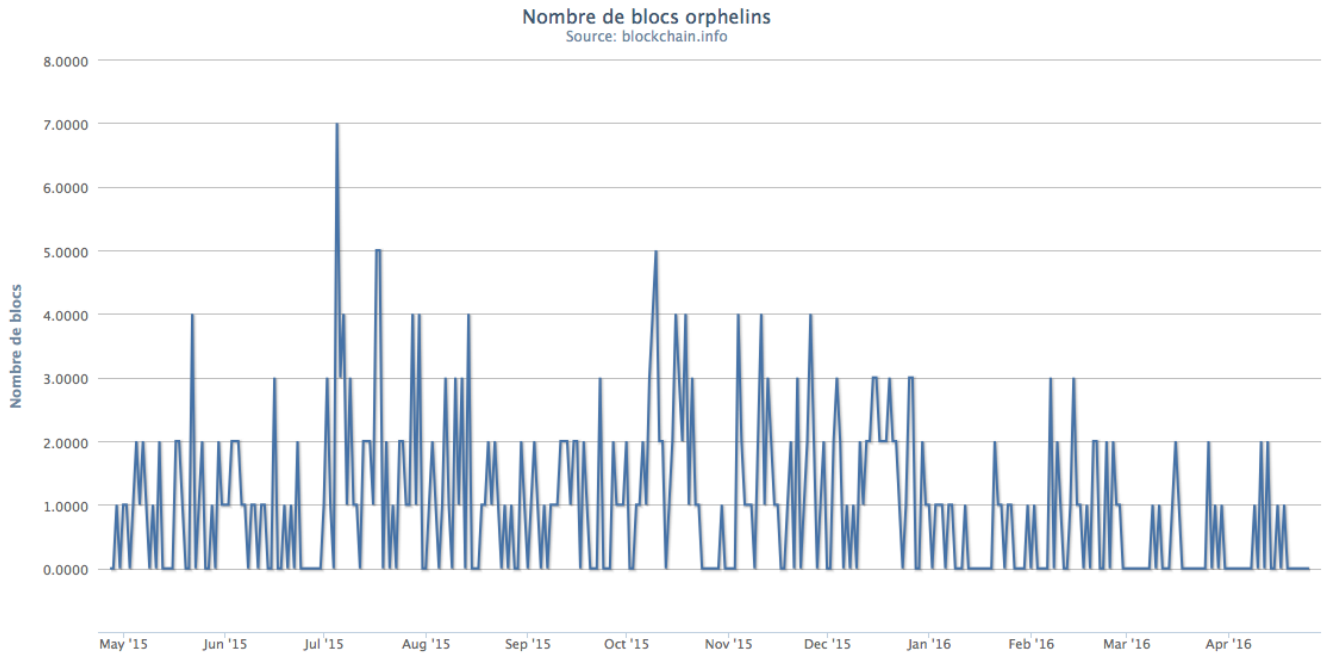
LIMOS LABORATOIRE D'INFORMATIQUE, DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

Miner = Validation des transactions

Cible: 00000000000000000254845fa930deac4086b3e3bce21147e93f463b206d8076



- La chaîne la plus longue persiste (attaque 51 %)
- Validation toutes les 10 minutes (6 confirmations)



LIMOS LABORAT DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

Autres crypto-monnaies

Classification des Altcoins

1. “Pourris coins”
2. Clônes de Bitcoin

3. Minage plus utiles, moins énergivores
4. Non-basés sur la preuve de travail
   - Proof of Stake (Peercoin)
   - Proof of Retreivability (Permacoin)
   - Proof of Capacity (Burstcoin)
   - Proof of Space (SpaceMint)

LIMOS

Bitcoin : Crypto-monnaie dématérialisée décentralisée

- Preuve de travail = Objectif de Hachage
- Création de la monnaie = récompense aux mineurs
- Miner = difficle + energivore

Bitcoin : Crypto-monnaie dématérialisée décentralisée

- Preuve de travail = Objectif de Hachage
- Création de la monnaie = récompense aux mineurs
- Miner = difficile + energivore

- Perte ou vol de la clef secrète = irréversible
- Monnaie anonyme et traçable

Quizz

1. (10 pts) Que signifie les acronymes suivants : RSA, MAC, OTP, CBC, AES, DES, DPO, RGPD, CNIL, CBC.
2. (1 pt) Dessiner une courbe eliptique.
3. (4 pts) Quelles sont les différences entre un chiffrement à clef publique et un chiffrement à clef symmétrique (1 pt par différence).
4. (3 points) Citez les 3 pilliers de l'IE.
5. (2 points) Expliquer l'attaque EFAII.

Thank you for your attention.

Questions ?