

# Security protocols - Introduction course

Véronique Cortier<sup>1</sup>  
cortier@loria.fr

# Roadmap of the course

Two parts in parallel :

- Security protocols (myself)
  - Course 1 : Introduction to security protocols
  - Course 2 : Model and analysis for security protocols : ProVerif
  - Course 3 : How does ProVerif work ?
  - Course 4 (?) : the SSL protocol and recent attacks
- Cryptography (Émmanuel Thomé)

# Évaluation du cours

Deux notes :

- Examen final le 24 novembre 2020 sur les 2 parties du cours
- Projet : championnat de protocoles (équipes de 2 à 3 personnes, envoyez moi le nom de l'équipe + emails avant le [23 septembre.](#))

# Évaluation du cours

Deux notes :

- Examen final le 24 novembre 2020 sur les 2 parties du cours
- Projet : championnat de protocoles (équipes de 2 à 3 personnes, envoyez moi le nom de l'équipe + emails avant le [23 septembre](#).)
  - Phase 1 : créez votre propre protocole
    - Plus votre protocole pèse lourd, plus vous partez avec un nombre négatif de points !
    - [Date limite : 7 octobre](#), 20 points de malus / jour de retard

# Évaluation du cours

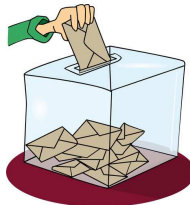
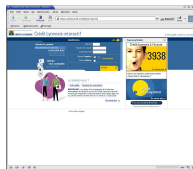
Deux notes :

- Examen final le 24 novembre 2020 sur les 2 parties du cours
- Projet : championnat de protocoles (équipes de 2 à 3 personnes, envoyez moi le nom de l'équipe + emails avant le [23 septembre](#).)
  - Phase 1 : créez votre propre protocole
    - Plus votre protocole pèse lourd, plus vous partez avec un nombre négatif de points !
    - [Date limite : 7 octobre](#), 20 points de malus / jour de retard
  - Phase 2 : attaquez les autres !
    - Tous les protocoles seront rendus publics 9 octobre
    - Fin des attaques le 6 novembre
  - Phase 3 (en parallèle à la phase 2) : modélisation de votre protocole + 2 autres au choix, en ProVerif ([15 novembre](#))
    - au moins un avec attaque
    - au moins un sans attaque

Context : cryptographic protocols

Cryptographic protocols are widely used in everyday life.

→ They aim at securing communications over public or insecure networks.





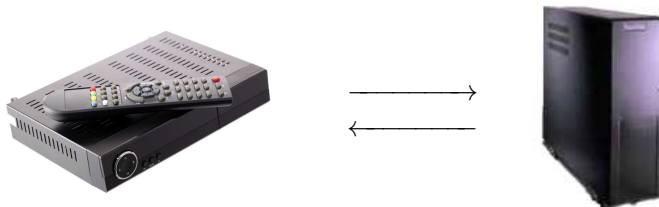
# Credit Card payment



- It is a real card ?
- Is the pin code protected ?

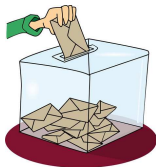


# Pay-per-view devices



- Checks your identity
- You should be granted access to the movie only once
- You should not be able to broadcast the movie to other people

# Electronic voting



As secure as paper ballot systems??

# Electronic purse



- It should not possible to add money without paying.
- It should not be possible to create fake electronic purse.

# Plan du 1er cours

- 1 Introduction on security protocols
  - Context
- 2 Terminologie
  - Les propriétés de sécurité
  - Les attaques typiques
  - Les primitives cryptographiques
- 3 Exemples d'attaques
- 4 Examples of security protocols
  - Commutative encryption (RSA)
  - Credit Card payment
  - Needham-Schroeder Example
- 5 Comment analyser un protocole ?
  - Vérification de logiciels critiques
  - Vérification de protocoles cryptographiques

# Les propriétés de sécurité

## Les propriétés de sécurité

# Authentification

## Definition (authentification)

Capacité à reconnaître de façon sûre l'identité d'une entité.

### Entités possibles :

- personne physique ou morale (individu ou société)
- ordinateur ou groupe d'ordinateurs (correspondant à une adresse IP par exemple)
- entité informatique (serveur par exemple)

# Confidentialité

## Definition (Confidentialité)

Une information ne peut être connue que par les entités habilitées

### Exemples :

- fichier sensible
- dossier médical
- mot de passe, code secret
- clefs privées

# Anonymat

## Definition (Anonymat)

Non identification de l'identité

### Exemples :

- téléphone portable
- visite de site web
- appartenance à un groupe



# Non-répudiation

## Definition (Non-répudiation)

Assure que l'auteur d'un acte ne peut ensuite dénier l'avoir effectuer.

On distingue couramment :

- **Non-répudiation d'origine** : l'émetteur d'un ordre ou d'un message ne peut nier l'émission
- **Non-répudiation de réception** : le récepteur d'un ordre ou d'un message ne peut l'avoir reçu.  
→ "Accusé de réception" de La Poste

# Intégrité

## Definition (Intégrité)

Assure qu'une information ne peut être modifiée, sauf par des entités habilitées (selon une politique de sécurité par exemple)

### Exemples :

- ordre de virement  
→ Il ne faut pas pouvoir modifier le montant
- Protection contre les virus informatiques : le code binaire des programme ne doit pas pouvoir être altéré.

# Les attaques typiques

## Les attaques typiques

# Déni de service

## Definition (Déni de service)

Attaque visant à nuire au fonctionnement correct, en terme de fonctionnalité ou de performances.

## Objectifs

- nuire à une institution (site web du Trésor Public, d'une grande compagnie, ...)
- Faire tomber un premier rideau de défense pour lancer une autre attaque

# Déguisement - Mascarade

## Definition (Déguisement - Mascarade)

Attaque contre une propriété d'authentification.

Consiste à usurper l'identité d'un sujet ou d'objet pour acquérir des droits illicites.

### Exemples :

- cartes bancaires
- passage root
- accès aux comptes informatiques de profs

# Rejeu (replay attacks)

## Definition (Rejeu)

Consiste à répéter une séquence d'actions sans la modifier ou renvoyer des messages déjà émis.

## Exemples :

- Relancer plusieurs fois le même ordre de crédit.
- Renvoyer le mot de passe chiffré utilisé par Alice pour s'authentifier auprès d'un serveur.

# Attaques visant la confidentialité des données

- Accès à des fichiers sensibles à l'aide d'une authentification frauduleuse.

# Attaques visant la confidentialité des données

- Accès à des fichiers sensibles à l'aide d'une authentification frauduleuse.
- Présence de **Canaux cachés** :

Fichier A (confidentiel)	oui/non
Fichier B (public)	oui/non

Si  $A = B$  print "Egaux" sinon print "Différents".



# Attaques visant la confidentialité des données

- Accès à des fichiers sensibles à l'aide d'une authentification frauduleuse.
- Présence de **Canaux cachés** :

Fichier A (confidentiel)	oui/non
Fichier B (public)	oui/non

Si  $A = B$  print "Egaux" sinon print "Différents".

- **Attaques par induction** par croisement de sources d'informations.  
**Exemple :**

- interrogation d'une base de données épidémiologique (non nominative)
- informations disponibles sur le web

# Attaques par canaux cachés

Un détour :

Comment calculer  $7^{42}$  ?

- 1 on calcule  $7 \times 7 \times \dots \times 7$   
→ 42 opérations

# Attaques par canaux cachés

Un détour :

Comment calculer  $7^{42}$  ?

- 1 on calcule  $7 \times 7 \times \dots \times 7$   
→ 42 opérations

- 2 **Exponentiation rapide**

On remarque que 42 s'écrit 10101 en binaire *i.e.*

$$42 = 2^5 + 2^3 + 2$$

- Par carrés successifs, on calcule  $7, 7^2, 7^{2^2}, 7^{2^3}, 7^{2^4}, 7^{2^5}$ .
- Puis on multiplie  $7^{42} = 7^{2^5} \times 7^{2^3} \times 7$
- Soit 5 carrés et 2 multiplications seulement !

# Chiffrement RSA

Chiffrement RSA de  $m$  par  $k$  : on calcule  $m^k \bmod n$  où

$$k \approx 2^{2048} \approx 10^{617}$$

Calcul de  $m^k$  : 2048 carrés et au plus 2048 multiplications.

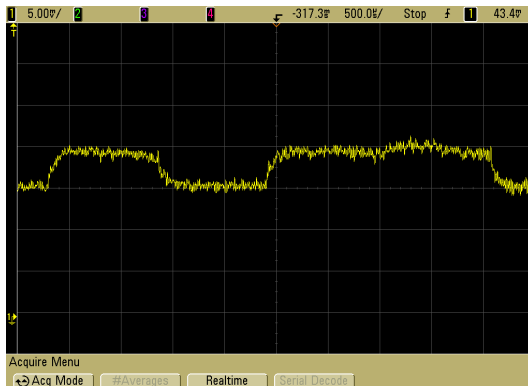
# Chiffrement RSA

Chiffrement RSA de  $m$  par  $k$  : on calcule  $m^k \bmod n$  où

$$k \approx 2^{2048} \approx 10^{617}$$

Calcul de  $m^k$  : 2048 carrés et au plus 2048 multiplications.

**Difficulté** : Un carré est plus "facile" qu'une multiplication



# Différents canaux cachés

- Mesure de consommation
- Temps d'exécution
- Bruit (composants, imprimante, ...)
- Rayonnement magnétique
- Longueurs des messages
- ...

# Attaques visant la confidentialité des données

- Accès à des fichiers sensibles à l'aide d'une authentification frauduleuse.

# Attaques visant la confidentialité des données

- Accès à des fichiers sensibles à l'aide d'une authentification frauduleuse.
- Présence de **Canaux cachés** :

Fichier A (confidentiel)	oui/non
Fichier B (public)	oui/non

Si  $A = B$  print "Egaux" sinon print "Différents".

- **Attaques par induction** par croisement de sources d'informations.

**Exemple :**

- interrogation d'une base de données épidémiologique (non nominative)
- informations disponibles sur le web



# Requêtes de recherche AOL

Août 2006 : AOL Research publie 3 mois de log des requêtes de plus de 650 000 utilisateurs.

HOME PAGE TODAY'S PAPER VIDEO MOST POPULAR U.S. Edition ▼

SUBSCRIBE NOW Log In Register Now Help

**The New York Times** Search All NYTimes.com Go

WORLD U.S. N.Y. / REGION BUSINESS TECHNOLOGY SCIENCE HEALTH SPORTS OPINION ARTS STYLE TRAVEL JOBS REAL ESTATE AUTOS

## A Face Is Exposed for AOL Searcher No. 4417749

By MICHAEL BARBARO and TOM ZELLER Jr.; Saul Hansell contributed reporting for this article  
Published: August 9, 2006

Buried in a list of 20 million Web search queries collected by AOL and recently released on the Internet is user No. 4417749. The number was assigned by the company to protect the searcher's anonymity, but it was not much of a shield.

No. 4417749 conducted hundreds of searches over a three-month period on topics ranging from "numb fingers" to "60 single men" to "dog that urinates on everything."

And search by search, click by click, the identity of AOL user No. 4417749 became easier to discern. There are queries for "landscapers in Lilburn, Ga," several people with the last name Arnold and "homes sold in shadow lake subdivision gwinnett county georgia."

It did not take much investigating to follow that data trail to Thelma Arnold, a 62-year-old widow who lives in Lilburn, Ga., frequently researches her friends' medical ailments and loves her three dogs. "Those are my searches," she said, after a reporter read part of the list to her.

AOL removed the search data from its site over the weekend and apologized for its release, saying it was an unauthorized move by a team that had hoped it would benefit academic

Facebook TWITTER GOOGLE+ EMAIL SHARE PRINT REPRINTS

MOST EMAILED RECOMMENDED FOR YOU

We don't have any personalized recommendations for you at this time. Please try again later.

Log in to discover more articles based on what you've read.

Log In Register Now What's This? | Don't Show

# Netflix

\$ 1 million Netflix Prize : concours pour améliorer le système de recommandation de Netflix, publication d'une partie des notes d'environ 500 000 utilisateurs (1/8 de la base de données seulement)



[A. Narayanan and V. Shmatikov, S&P'08] : Un algorithme efficace pour casser l'anonymat

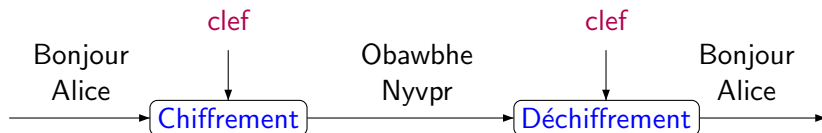
- Deux entrées différentes sont toujours à moins de 50% semblables
- rapprochement avec la base (publique) IMDb

# Les primitives cryptographiques

## Les primitives cryptographiques

# Chiffrement symétrique

Notation :  $\{m\}_k$



On utilise la même clef pour chiffrer et déchiffrer.

# Avantages et inconvénients

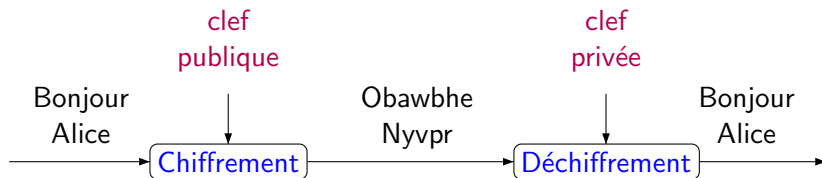
- Chiffrement efficace  
→ permet de chiffrer très rapidement des messages de tailles importantes
- Il faut partager une clef secrète entre deux clients
  - Utilisation d'un serveur de confiance
  - Recours à des protocoles d'établissement de clefs

Exemples : DES, triple DES, AES, ...

# Chiffrement asymétrique

Clef asymétrique :  $pk(A)$

Notation :  $\{m\}_{pk(A)}$



On chiffre avec la **clef publique** et on déchiffre avec la **clef privée**.

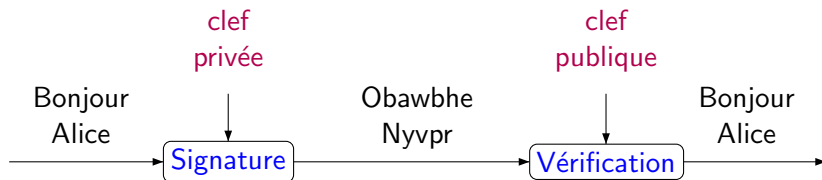
# Avantages et inconvénients

- Chiffrement plus coûteux en temps de calcul
- Ne demande pas de **connaissance préalable** entre les deux clients.
  - à la base de nombreux protocoles d'établissement de clefs
- Il faut **authentifier** les clefs publiques

Exemples : RSA, El Gamal, OAEP+

# Signature

- Principe inverse du chiffrement à clef publique.
- Tout le monde connaît la **clef de vérification**  $vk(A)$ .
- La **clef de signature**  $sk(A)$  est secrète.
- Notation :  $[m]_{sk(A)}$





# Propriétés des signatures

## Authentification et non répudiation

- La signature ne peut être imitée. Elle authentifie le signataire.

# Propriétés des signatures

## Authentification et non répudiation

- La signature ne peut être imitée. Elle authentifie le signataire.
- La signature appartient à un seul document.  
→ on ne peut pas découper une signature sur un document et la recoller sur un autre.

# Propriétés des signatures

## Authentification et non répudiation

- La signature ne peut être imitée. Elle authentifie le signataire.
- La signature appartient à un seul document.  
→ on ne peut pas découper une signature sur un document et la recoller sur un autre.
- Le document signé ne peut être modifié, même partiellement.

# Propriétés des signatures

## Authentification et non répudiation

- La signature ne peut être imitée. Elle authentifie le signataire.
- La signature appartient à un seul document.  
→ on ne peut pas découper une signature sur un document et la recoller sur un autre.
- Le document signé ne peut être modifié, même partiellement.
- La signature peut être contrôlée par un tiers.

# Propriétés des signatures

## Authentification et non répudiation

- La signature ne peut être imitée. Elle authentifie le signataire.
- La signature appartient à un seul document.  
→ on ne peut pas découper une signature sur un document et la recoller sur un autre.
- Le document signé ne peut être modifié, même partiellement.
- La signature peut être contrôlée par un tiers.
- La signature ne peut être reniée.

# Fonctions de hachage

## Definition (Fonction de hachage)

Fonction qui à un message  $M$  de longueur quelconque, fait correspondre un message  $h(M)$  de longueur fixe, appelé **hache**, **résumé** ou **empreinte numérique**.

$$h : M \mapsto h(M)$$

## Applications :

- Économie de place (on signe le hache d'un message plutôt que le message tout entier).
- Protection pour l'intégrité des messages envoyés.
- La partie redondante d'un code correcteur d'erreur est une fonction de hachage.

# Propriétés des fonctions de hachage

Une fonction de hachage **n'est pas injective** puisqu'elle envoie les éléments d'un ensemble dans un ensemble plus petit.

On voudrait cependant qu'une fonction de hachage soit **calculatoirement** injective.

# Propriétés des fonctions de hachage

Une fonction de hachage **n'est pas injective** puisqu'elle envoie les éléments d'un ensemble dans un ensemble plus petit.

On voudrait cependant qu'une fonction de hachage soit **calculatoirement** injective.

- **Collision faible difficile**

Si le problème, étant donné  $M$ , de trouver  $M'$  tel que  $h(M) = h(M')$ , est difficile (au moins super-polynomiale en fonction du cardinal de l'ensemble des messages).

- **Collision forte difficile**

Si le problème, de trouver  $M$  et  $M'$  tel que  $h(M) = h(M')$ , est difficile (au moins super-polynomiale en fonction du cardinal de l'ensemble des messages).



# Génération de nombres aléatoires

Les nombres aléatoires sont aussi appelés **nonces**, notés

$N, N_1, N_2, \dots, N_a, N_b, \dots$

- De nombreuses données doivent avoir des valeurs **imprédictibles**.
- La connaissance d'une partie de la suite ne doit pas permettre de prédire son évolution.

Deux principales sources d'aléas :

- Physique ("vrais" nombres aléatoires)
- Informatique (nombres pseudo aléatoires)

# Quelques attaques logiques

## Quelques attaques logiques

→ Sans même attaquer les primitives !

# Quel attaquant ?

On souhaite des protocoles sûrs même en présence d'un **attaquant**

- qui peut **lire** les messages envoyés sur le réseau,
- qui peut **intercepter et envoyer** de nouveaux messages.
- qui peut **prendre part** au protocole sous ses propres identités (mail, FaceBook, etc.)



# Échange de secret



On peut utiliser la clef  $k = 785341006$

Mon code bancaire est  $\{3443\}_{785341006}$



La clef est 785341006

Je déchiffre...

le code secret est 3443

# Attaques sur les décodeurs

Kentucky Fried Chip : bloquer l'ordre de désabonnement

Renvoi de messages :



$n^{\circ} \text{ carte}, \{c_1, \dots, c_k\}_{K_{Cs}}$

$n^{\circ} \text{ carte}, \{c'_1, \dots, c'_k\}_{K_{Cs}}$





## How does this work ?

A cryptographic protocol :

**Protocol** describes how each participant should behave in order to get e.g. a common key.

**Cryptographic** makes uses of cryptographic primitives (e.g. encryption, signatures, hashes, ...)

## How to exchange a secret with commutative encryption

First : a small challenge for your nephews / nieces / cousins / children.



# A completely fictitious town

Two types of inhabitants :

**Sedentary inhabitants** stay at their home

**Post office workers** deliver boxes between sedentary inhabitants

# A completely fictitious town

Two types of inhabitants :

**Sedentary inhabitants** stay at their home

**Post office workers** deliver boxes between sedentary inhabitants

**Axiom 1** Post office workers may steal any unlocked box  
(Reminder : this scenario is entirely fictitious!)

**Axiom 2** The content of locked boxes CANNOT be theft.

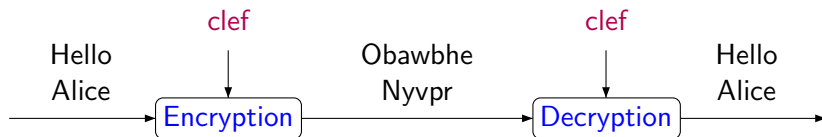


## Challenge

How Alice (sedentary) can send a gift to Bob (also sedentary) ?

# Commutative Symmetric encryption

Symmetric encryption, denoted by  $\{m\}_k$



The same key is used for **encrypting** and **decrypting**.

Commutative (symmetric) encryption (e.g. RSA)

$$\{\{m\}_{k_1}\}_{k_2} = \{\{m\}_{k_2}\}_{k_1}$$

# Exchanging a secret with commutative encryption (RSA)

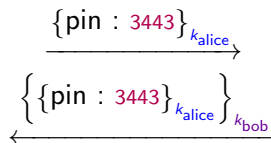


$\{\text{pin} : 3443\}_{k_{\text{alice}}}$

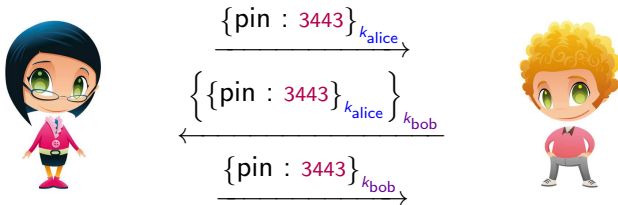
→



# Exchanging a secret with commutative encryption (RSA)

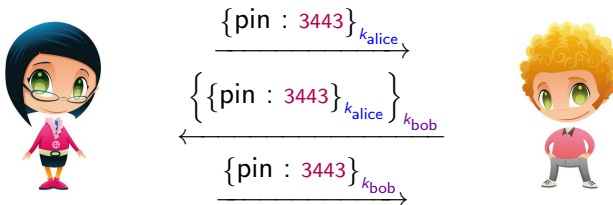


# Exchanging a secret with commutative encryption (RSA)



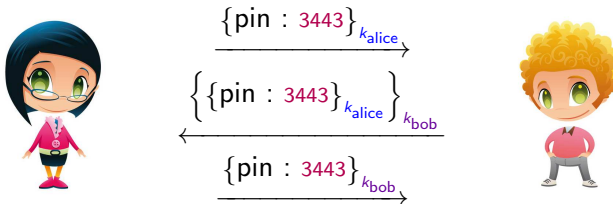
$$\text{Since } \left\{ \left\{ \text{pin} : 3443 \right\}_{k_{\text{alice}}} \right\}_{k_{\text{bob}}} = \left\{ \left\{ \text{pin} : 3443 \right\}_{k_{\text{bob}}} \right\}_{k_{\text{alice}}}$$

# Exchanging a secret with commutative encryption (RSA)

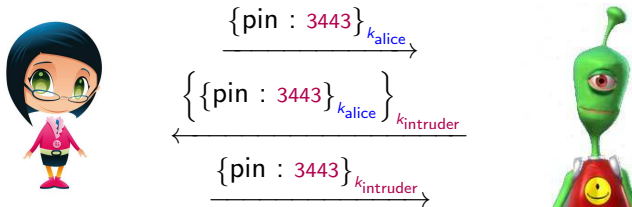


→ It does not work! (Authentication problem)

# Exchanging a secret with commutative encryption (RSA)



→ It does not work ! (Authentication problem)





# Credit Card payment



- It is a real card ?
- Is the pin code protected ?

# Behavior in the usual case



- The waiter introduces the credit card.
- The waiter enters the amount  $m$  of the transaction on the terminal.
- The terminal authenticates the card.
- The customer enters his secret code.  
If the amount  $m$  is greater than 100 euros  
(and in only 20% of the cases)
  - The terminal asks the bank for authentication of the card.
  - The bank provides authentication.

# More details

4 actors : Bank, Customer, Card and Terminal.

Bank owns

- a signing key  $K_B^{-1}$ , secret,
- a verification key  $K_B$ , public,
- a secret symmetric key for each credit card  $K_{CB}$ , secret.

Card owns

- Data : last name, first name, card's number, expiration date,
- Signature's Value  $VS = \{hash(Data)\}_{K_B^{-1}}$ ,
- secret key  $K_{CB}$ .

Terminal owns the verification key  $K_B$  for bank's signatures.

# Credit card payment Protocol (in short)

The terminal reads the card :

$$1. \quad C_a \rightarrow T : \text{Data}, \{\text{hash}(\text{Data})\}_{K_B^{-1}}$$

# Credit card payment Protocol (in short)

The terminal reads the card :

$$1. \quad Ca \rightarrow T : \text{Data}, \{\text{hash}(\text{Data})\}_{K_B^{-1}}$$

The terminal asks for the secret code :

$$2. \quad T \rightarrow Cu : \text{secret code?}$$

$$3. \quad Cu \rightarrow Ca : 1234$$

$$4. \quad Ca \rightarrow T : \text{ok}$$

# Credit card payment Protocol (in short)

The terminal reads the card :

$$1. \quad Ca \rightarrow T : \text{Data}, \{hash(\text{Data})\}_{K_B^{-1}}$$

The terminal asks for the secret code :

$$2. \quad T \rightarrow Cu : \text{secret code?}$$

$$3. \quad Cu \rightarrow Ca : 1234$$

$$4. \quad Ca \rightarrow T : ok$$

The terminal calls the bank :

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

$$6. \quad B \rightarrow T : N_b$$

$$7. \quad T \rightarrow Ca : N_b$$

$$8. \quad Ca \rightarrow T : \{N_b\}_{K_{CB}}$$

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

$$10. \quad B \rightarrow T : ok$$

# Some flaws

The security was initially ensured by :

- the cards were very difficult to reproduce,
- the protocol and the keys were secret.

But

- cryptographic flaw : 320 bits keys can be broken (1988),
- logical flaw : no link between the secret code and the authentication of the card,
- fake cards can be build.

# Some flaws

The security was initially ensured by :

- the cards were very difficult to reproduce,
- the protocol and the keys were secret.

But

- cryptographic flaw : 320 bits keys can be broken (1988),
- logical flaw : no link between the secret code and the authentication of the card,
- fake cards can be build.

→ “YesCard” build by Serge Humpich  
(1998 in France).



# How does the “YesCard” work ?

## Logical flaw

1.  $Ca \rightarrow T : \text{Data}, \{\text{hash}(\text{Data})\}_{K_B^{-1}}$
2.  $T \rightarrow Ca : \text{secret code?}$
3.  $Cu \rightarrow Ca : 1234$
4.  $Ca \rightarrow T : \text{ok}$

# How does the “YesCard” work ?

## Logical flaw

1.  $Ca \rightarrow T$  :  $\text{Data}, \{\text{hash}(\text{Data})\}_{K_B^{-1}}$
2.  $T \rightarrow Ca$  : *secret code?*
3.  $Cu \rightarrow Ca'$  : 2345
4.  $Ca' \rightarrow T$  : *ok*

# How does the “YesCard” work ?

## Logical flaw

1.  $C_a \rightarrow T$  :  $\text{Data}, \{\text{hash}(\text{Data})\}_{K_B^{-1}}$
2.  $T \rightarrow C_a$  : *secret code?*
3.  $C_u \rightarrow C_a'$  : 2345
4.  $C_a' \rightarrow T$  : *ok*

**Remark :** there is always somebody to debit.

→ creation of a fake card

# How does the “YesCard” work ?

## Logical flaw

1.  $Ca \rightarrow T$  : Data,  $\{hash(Data)\}_{K_B^{-1}}$
2.  $T \rightarrow Ca$  : *secret code?*
3.  $Cu \rightarrow Ca'$  : 2345
4.  $Ca' \rightarrow T$  : *ok*

**Remark :** there is always somebody to debit.

→ creation of a fake card

1.  $Ca' \rightarrow T$  : XXX,  $\{hash(XXX)\}_{K_B^{-1}}$
2.  $T \rightarrow Cu$  : *secret code?*
3.  $Cu \rightarrow Ca'$  : 0000
4.  $Ca' \rightarrow T$  : *ok*

# Another example

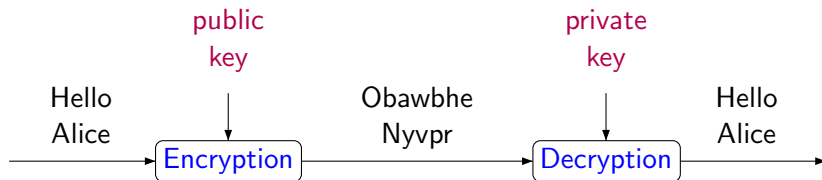
The “famous” Needham-Schroeder public key  
protocol

(and its associated **Man-In-The-Middle Attack**)

# Public key encryption

Public key :  $pk(A)$

Encryption :  $\{m\}_{pk(A)}$



Encryption with the **public key** and decryption with the **private key**.

Invented only in the late 70's!

# Needham-Schroeder public key protocol

$N_a$  Random number (called nonce) generated by A.

$N_b$  Random number (called nonce) generated by B.



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



# Needham-Schroeder public key protocol

$N_a$  Random number (called nonce) generated by A.

$N_b$  Random number (called nonce) generated by B.



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





# Needham-Schroeder public key protocol

$N_a$  Random number (called nonce) generated by A.

$N_b$  Random number (called nonce) generated by B.



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

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

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



# Needham-Schroeder public key protocol

$N_a$  Random number (called nonce) generated by  $A$ .

$N_b$  Random number (called nonce) generated by  $B$ .



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


## Questions :

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

# Needham-Schroeder public key protocol

$N_a$  Random number (called nonce) generated by  $A$ .

$N_b$  Random number (called nonce) generated by  $B$ .



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


## Questions :

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

→ An attack was discovered in 1996, 17 years after the publication of the protocol!

## Needham-Schroeder Example

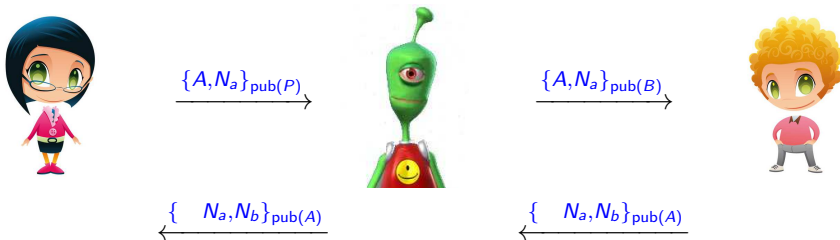
## Man in the middle attack


$$\xrightarrow{\{A, N_a\}_{\text{pub}(P)}}$$

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

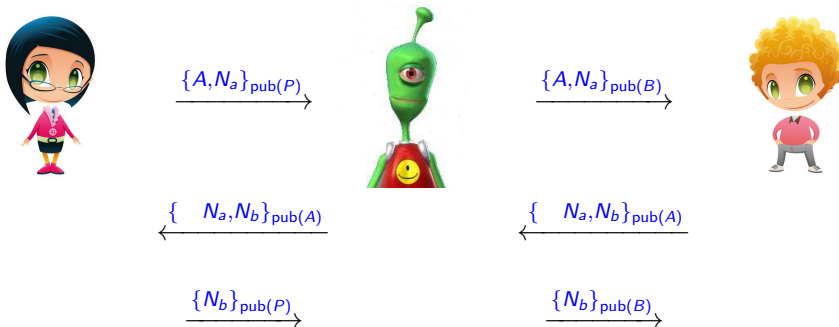

## Needham-Schroeder Example

## Man in the middle attack



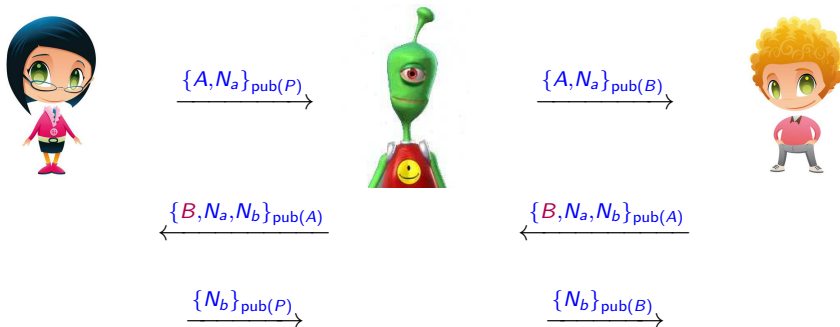
## Needham-Schroeder Example

## Man in the middle attack



## Needham-Schroeder Example

## Man in the middle attack



Fixing the flaw : add the identity of  $B$ .

# Analyse des protocoles cryptographiques

- 1 L'analyse de logiciels critiques en général.
- 2 Le cas des protocoles cryptographiques.



# Fusée Ariane 5



11 décembre 2002 :  
vol inaugural

Un crash dû ... à un bug logiciel !

# Sonde Mars Climate Orbiter



26 septembre 1999 : perte de la sonde due ...  
... à une erreur de conversion !

# Comment fait-on ?

## 1 Tests

- à la main ou génération automatique
- vérification d'un nombre **fini** de comportements

# Comment fait-on ?

## 1 Tests

- à la main ou génération automatique
- vérification d'un nombre **fini** de comportements

## 2 Vérification (preuve formelle)

- à la main ou à l'aide d'ordinateur
- vérification de **tous les comportements possibles**
- plus difficile

# Airbus A380

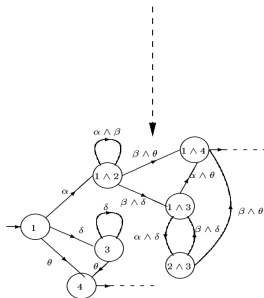


500 000 lignes de code, plus gros programme jamais vérifié  
(Le Monde 26 avril 2005)

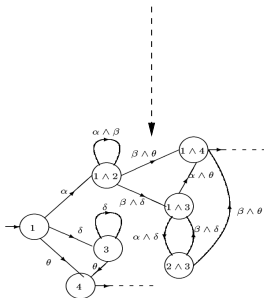
# La vérification formelle



## 00



# La vérification formelle



?  
=

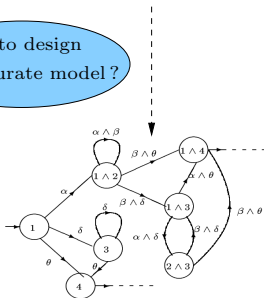
$\emptyset$



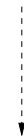
# La vérification formelle



How to design  
an accurate model?



?

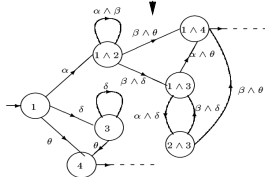


$\emptyset$

# La vérification formelle



How to design  
an accurate model?



?

$\models$

How to check  
the property?

$\emptyset$

# Deux grandes étapes

**Modélisation** Donner une signification formelle / mathématique :

- au programme, à l'application d'une part
- à la propriété souhaitée “pas de bugs”

**Vérification** Comment fait-on pour montrer qu'une application vérifie la propriété souhaitée ?

- techniques de preuves
- outils automatiques
- décidabilité des propriétés étudiées

# Application aux protocoles cryptographiques

## Un important besoin de sécurité

- Tester le protocole pour des attaques connues ne suffit pas.

# Application aux protocoles cryptographiques

## Un important besoin de sécurité

- Tester le protocole pour des attaques connues ne suffit pas.
- Corriger un protocole après son déploiement coûte cher

# Application aux protocoles cryptographiques

## Un important besoin de sécurité

- Tester le protocole pour des attaques connues ne suffit pas.
- Corriger un protocole après son déploiement coûte cher
- Les normes actuelles de sécurité requièrent des méthodes formelles (=preuves)  
e.g. **Evaluation Assurance Levels** définis par l'industrie :
  - EAL5 : semiformally designed and tested
  - EAL6 : semiformally verified design and tested
  - EAL7 (niveau le + haut) : formally verified design and tested

# Difficultés de la vérification

## Intrus

- intercepte les messages,
- analyse et crée de nouveaux messages,
- envoie des messages,
- a une mémoire arbitraire.

# Difficultés de la vérification

## Intrus

- intercepte les messages,
- analyse et crée de nouveaux messages,
- envoie des messages,
- a une mémoire arbitraire.

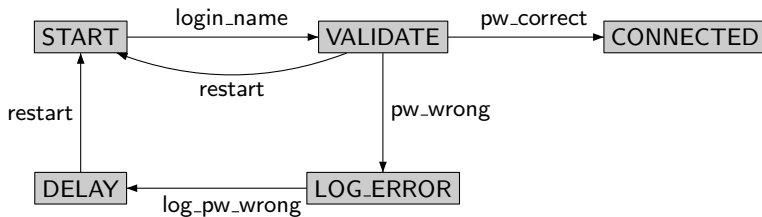
## Protocole

- nombre arbitraire de participants,
- nombre arbitraire de sessions,
- taille arbitraire des messages.

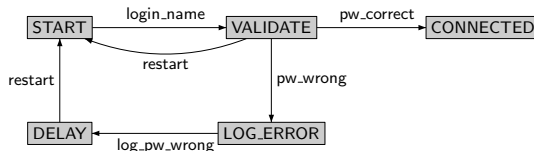


# Une première approche

Pourquoi ne pas modéliser un protocole à l'aide d'un automate ?



# Comment modéliser un protocole de sécurité ?

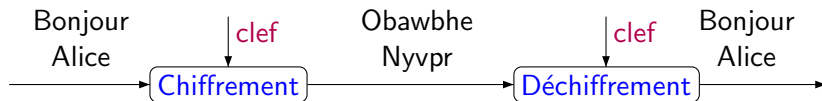


- La réponse de chaque participant **dépend étroitement des données reçues**.
- A chaque étape, un attaquant peut **créer des messages arbitraires**.
- Le comportement de l'attaquant **dépend étroitement de ce qu'il a pu déjà observer**.

→ Il est important d'avoir une modélisation fine des messages.

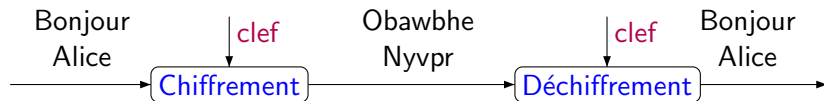
# Comment modéliser le chiffrement ?

Le chiffrement est une boîte noire :  $\{-\}_-$



# Comment modéliser le chiffrement ?

Le chiffrement est une boîte noire :  $\{-\}_-$



On suppose que le chiffrement est parfait :

On ne peut rien apprendre de  $\{m\}_k$  à moins de connaître  $k$ .

## Some bibliographical references (1/4)

A book that gathers several chapters on models and procedures for security protocols

*Formal Models and Techniques for Analyzing Security Protocols.*  
Cryptography and Information Security Series 5, IOS Press, 2011.

Notes of a Master course (in French)

<http://www.loria.fr/~cortier/Cours/cours2010.pdf>

### On protocols

- B. Schneier. *Applied Cryptography Second Edition : protocols, algorithms, and source code in C*, J. Wiley & Sons, Inc. publisher, 1996.
- A. J. Menezes and P. C. van Oorschot and S. A. Vanstone. *Handbook of applied cryptography*, CRC Press publisher, 1997.

## Some bibliographical references (2/4)

### Procedures on Horn clauses, for an unbounded number of sessions

- B. Blanchet. *An Efficient Cryptographic Protocol Verifier Based on Prolog Rules*. CSFW 2001.
- H. Comon, V. Cortier. *Tree automata with one memory set constraints and cryptographic protocols*. Theoretical Computer Science 2005.

### Constraint solving for a bounded number of sessions

- J. K. Millen, V. Shmatikov. *Constraint solving for bounded-process cryptographic protocol analysis*. ACM Conference on Computer and Communications Security 2001.
- H. Comon-Lundh, V. Shmatikov. *Intruder Deductions, Constraint Solving and Insecurity Decision in Presence of Exclusive or*. LICS 2003

# Some bibliographical references (3/4)

## Voting protocols

- S. Delaune, S. Kremer, M. Ryan. *Verifying privacy-type properties of electronic voting protocols*. Journal of Computer Security 2009.
- V. Cortier, B. Smyth. *Attacking and Fixing Helios : An Analysis of Ballot Secrecy*. CSF 2011.

## Static equivalence

- S. Ciobaca, S. Delaune, S. Kremer. *Computing Knowledge in Security Protocols Under Convergent Equational Theories*. J. Autom. Reasoning 2012.
- M. Baudet, V. Cortier, and S. Delaune. *YAPA : A generic tool for computing intruder knowledge*. ACM Transactions on Computational Logic, 2012.

# Some bibliographical references (4/4)

## Soundness of formal models

- D. Micciancio, B. Warinschi. *Soundness of Formal Encryption in the Presence of Active Adversaries*. TCC 2004.
- V. Cortier, B. Warinschi. *Computationally Sound, Automated Proofs for Security Protocols*. ESOP 2005.
- H. Comon-Lundh, V. Cortier. *Computational soundness of observational equivalence*. ACM CCS 2008.