

# Verification of cryptographic protocols

Stéphanie Delaune

LSV, ENS de Cachan & CNRS & INRIA Saclay Île-de-France (project Secsi)

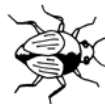
November 16, 2010

—→ accroître notre confiance dans les logiciels critiques

—→ accroître notre confiance dans les logiciels critiques

- **logiciel** : texte relativement long écrit dans un langage spécifique et qui sera **exécuté par un ordinateur**
- **critique** : une défaillance peut avoir des **conséquences désastreuses** en termes humains ou économiques

Ennemi public numéro 1 : le **bug** ...



Ennemi public numéro 1 : le **bug** ...



... aussi connu sous le nom de **bogue** !

Dans la vie quotidienne !

# Dans la vie quotidienne !



# Ariane V - 4 juin 1996



Un crash après 40 secondes de vol dû

...



Un crash après 40 secondes de vol dû  
...  
à un **bug logiciel** !

- 1 189 vols réussis pour Ariane IV,
- 2 réutilisation du logiciel de lancement d'Ariane IV,
- 3 ajout du nécessaire pour la nouvelle fusée.

→ Le logiciel d'Ariane IV contenait un bug !

# Sonde Mars Climate Orbiter - 26 septembre 1999



Perte de la sonde due ...

# Sonde Mars Climate Orbiter - 26 septembre 1999



Perte de la sonde due ... à un problème d'**unité de mesure** !

# Carte bancaire

La carte bleue est protégée par un grand nombre public dont on ne connaît pas la **factorisation**.



# Carte bancaire

La carte bleue est protégée par un grand nombre public dont on ne connaît pas la **factorisation**.



Nombre de 96 chiffres

213598703592091008239502270499962879705109534182641740644252  
4165008583957746445088405009430865999

# Carte bancaire

La carte bleue est protégée par un grand nombre public dont on ne connaît pas la factorisation.



## Nombre de 96 chiffres

213598703592091008239502270499962879705109534182641740644252  
4165008583957746445088405009430865999

## Affaire Serge Humpich (1997)

il factorise ce nombre de 96 chiffres et conçoit de fausses cartes bleues (les " YesCard ").

# Carte bancaire

La carte bleue est protégée par un grand nombre public dont on ne connaît pas la **factorisation**.



## Nombre de 96 chiffres

213598703592091008239502270499962879705109534182641740644252  
4165008583957746445088405009430865999

## Affaire Serge Humpich (1997)

il factorise ce nombre de 96 chiffres et conçoit de fausses cartes bleues (les " **YesCard** ").

→ Depuis, le nombre utilisé pour sécuriser les cartes bancaires comportent **232 chiffres**.

→ une petite modification (quelques caractères) peut le transformer complètement.

## Un besoin crucial de vérification

- pour des **raisons économiques**  
→ Ariane 5, carte bancaire, ...
- mais parfois il y a aussi des **vies humaines** en jeu  
→ la machine Therac-25 dans les années 80  
→ **logiciels embarqués** dans les voitures, les avions, ...
- enjeux **démocratiques**  
→ vote électronique



## Tests

- à la main ou génération automatique ;
- vérification d'un **nombre fini** de cas.



## Tests

- à la main ou génération automatique ;
- vérification d'un **nombre fini** de cas.

∞ ∞ ∞

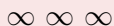
Accéder à l'**infini** : un rêve impossible ?

∞ ∞ ∞

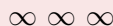


## Tests

- à la main ou génération automatique ;
- vérification d'un **nombre fini** de cas.



Accéder à l'**infini** : un rêve impossible ?



## Vérification (preuves formelles)

→ preuves mathématiques

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



# Comment vérifier ces programmes ?

Les mathématiques et l'informatique à la rescousse !

## Les mathématiques et l'informatique à la rescousse !

Notre but :

- 1 faire des preuves mathématiques rigoureuses,
- 2 d'une façon automatique.

“Construire une machine à détecter les bugs”

# Comment vérifier ces programmes ?

## Les mathématiques et l'informatique à la rescousse !

Notre but :

- 1 faire des preuves mathématiques rigoureuses,
- 2 d'une façon automatique.

“Construire une machine à détecter les bugs”

1936 : une telle machine n'existe pas (Alan Turing)

... même dans le cas particulier des protocoles cryptographiques.



# Mais alors, que faisons nous ?

Le problème n'a pas de solution ....

# Mais alors, que faisons nous ?

Le problème n'a **pas de solution** ....  
mais seulement dans le **cas général**



# Mais alors, que faisons nous ?

Le problème n'a **pas de solution** ....  
mais seulement dans le **cas général**



Différentes pistes :

- résoudre le problème dans de nombreux **cas intéressants**,

# Mais alors, que faisons nous ?

Le problème n'a **pas de solution** ....  
mais seulement dans le **cas général**



Différentes pistes :

- résoudre le problème dans de nombreux **cas intéressants**,
- proposer des **procédures approchées**,

**Exemple** : si le vérificateur répond “**oui**” alors le logiciel est **sûr**, sinon on ne peut rien dire

# Outline of the talk

## 1 Introduction

## 2 Cryptographic protocols

- How protocols can be attacked ?
- How protocols can be proved secure ?

## 3 Towards more realism

- Algebraic properties
- Composition

## 4 Conclusion

# Cryptographic protocols vs (classical) programs

Some specificities :

- protocol are executed in an **hostile** environment
  - a powerful attacker who controls the communication network
- **unbounded** number of sessions running concurrently
- the **cryptographic primitives** play an important role
  - we have to take them into account.

# Outline of the talk

- 1 Introduction
- 2 Cryptographic protocols
  - How protocols can be attacked ?
  - How protocols can be proved secure ?
- 3 Towards more realism
  - Algebraic properties
  - Composition
- 4 Conclusion

# How cryptographic protocols can be attacked ?

## Breaking encryption



- Ciphertext-only attack,
- Known-plaintext attack, ...

## Logical attack





## Les challenges RSA

- défis lancés par le laboratoire RSA Security
- récompenses importantes offertes



## Les challenges RSA

- défis lancés par le laboratoire RSA Security
- récompenses importantes offertes

RSA-576	174 chiffres	réussi	2003
RSA-640	193 chiffres	réussi	2005
RSA-704	212 chiffres	non résolu – 30 000 dollars	
...	..	...	
RSA-2048	617 chiffres	non résolu – 200 000 dollars	

# Casser le chiffrement RSA



## Les challenges RSA

- défis lancés par le laboratoire RSA Security
- récompenses importantes offertes

RSA-576	174 chiffres	réussi	2003
RSA-640	193 chiffres	réussi	2005
RSA-704	212 chiffres	non résolu – 30 000 dollars	
...	..	...	
RSA-2048	617 chiffres	non résolu – 200 000 dollars	

→ Ces challenges ont été retirés en 2007 !

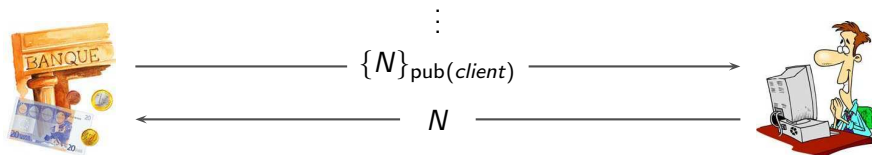
# Logical attack - What is it ?

## Electronic bank transfer



# Logical attack - What is it ?

## Electronic bank transfer



# Logical attack - What is it ?

## Electronic bank transfer



—  $\{N\}_{\text{pub}(\text{client})}$  →



# Logical attack - What is it ?

## Electronic bank transfer



# Logical attack - What is it ?

## Electronic bank transfer



# Logical attack - What is it ?

## Electronic bank transfer



## Logical attacks

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

# Outline of the talk

## 1 Introduction

## 2 Cryptographic protocols

- How protocols can be attacked ?
- How protocols can be proved secure ?

## 3 Towards more realism

- Algebraic properties
- Composition

## 4 Conclusion

# Cryptographic models

Main features :

- Messages are **bitstrings**
- Protocols are programs that exchange messages
- **Real** algorithms for cryptographic primitives
- **Powerful attacker** : any probabilistic polynomial time Turing machine

→ quite **realistic** model

# Cryptographic models

Main features :

- Messages are **bitstrings**
- Protocols are programs that exchange messages
- **Real** algorithms for cryptographic primitives
- **Powerful attacker** : any probabilistic polynomial time Turing machine

→ quite **realistic** model

**Advantage** : Clear and quite strong security guarantee

**Drawback** : Proofs are difficult, tedious and error-prone.

Main features :

- Messages are **abstracted by terms** (abstract objects)
- Protocols are programs that exchange messages
- Cryptographic primitives are **abstracted by function symbols**
- **Idealized attacker** : in particular, we have to describe what he can do.

→ very **abstract** model

Main features :

- Messages are **abstracted by terms** (abstract objects)
- Protocols are programs that exchange messages
- Cryptographic primitives are **abstracted by function symbols**
- **Idealized attacker** : in particular, we have to describe what he can do.

→ very **abstract** model

**Advantage** : Security proofs are easier to do and they can be mechanized

**Drawback** : the security guarantees obtained are rather unclear.

## Computational soundness

Computational soundness aims to establish sufficient conditions under which results obtained using symbolic models imply security under computational models.

→ Seminal paper : [Abadi & Rogaway, 2001](#)

Many other papers have been obtained in this area.

A survey is available [[Cortier et al., JAR 2010](#)]

Messages are abstracted by terms

- pairing  $\langle m_1, m_2 \rangle$ ,
- symmetric  $\text{enc}(m, k)$  and public key encryption  $\text{aenc}(m, \text{pub}(A))$ ,
- signature  $\text{sign}(m, \text{priv}(A))$ .

# Symbolic model

Messages are abstracted by terms

- pairing  $\langle m_1, m_2 \rangle$ ,
- symmetric  $enc(m, k)$  and public key encryption  $aenc(m, pub(A))$ ,
- signature  $sign(m, priv(A))$ .

Presence of an idealized attacker

- may **read**, **intercept** and **send** messages,
- may **build** new messages following **deduction rules** (symbolic manipulation on terms).



# Symbolic model

Messages are abstracted by terms

- pairing  $\langle m_1, m_2 \rangle$ ,
- symmetric  $enc(m, k)$  and public key encryption  $aenc(m, pub(A))$ ,
- signature  $sign(m, priv(A))$ .

Presence of an idealized attacker

- may **read**, **intercept** and **send** messages,
- may **build** new messages following **deduction rules** (symbolic manipulation on terms).



Examples :

$$\frac{m \quad k}{senc(m, k)}$$

$$\frac{senc(m, k) \quad k}{m}$$

$$\frac{aenc(m, pub(a)) \quad priv(a)}{m}$$

# Difficulties of the verification

Presence of an attacker ...



# Difficulties of the verification

Presence of an attacker ...

who controls the communication network :

- may **read** every message sent on the network
- may **intercept** and **send** new messages



# Difficulties of the verification

Presence of an attacker ...

who controls the communication network :

- may **read** every message sent on the network
- may **intercept** and **send** new messages



who has deduction capabilities

- encryption, decryption if he knows the decryption key,
- pairing, projection

# Difficulties of the verification

Presence of an attacker ...

who controls the communication network :

- may **read** every message sent on the network
- may **intercept** and **send** new messages



who has deduction capabilities

- encryption, decryption if he knows the decryption key,
- pairing, projection

Secrecy problem for an **unbounded** number of sessions is **undecidable**.

# Difficulties of the verification

Presence of an attacker ...

who controls the communication network :

- may **read** every message sent on the network
- may **intercept** and **send** new messages



who has deduction capabilities

- encryption, decryption if he knows the decryption key,
- pairing, projection

Secrecy problem for a **fixed** number of sessions is **decidable**.

# Needham-Schroeder's Protocol (1978)



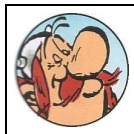
- $A \rightarrow B : \{\langle A, N_a \rangle\}_{\text{pub}(B)}$   
 $B \rightarrow A : \{\langle N_a, N_b \rangle\}_{\text{pub}(A)}$   
 $A \rightarrow B : \{N_b\}_{\text{pub}(B)}$



# Needham-Schroeder's Protocol (1978)



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



# Needham-Schroeder's Protocol (1978)



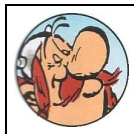
$A \rightarrow B : \{\langle A, N_a \rangle\}_{\text{pub}(B)}$   
 $B \rightarrow A : \{\langle N_a, N_b \rangle\}_{\text{pub}(A)}$   
•  $A \rightarrow B : \{N_b\}_{\text{pub}(B)}$



# Needham-Schroeder's Protocol (1978)



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



# Needham-Schroeder's Protocol (1978)


$$\begin{array}{l} A \rightarrow B : \quad \{\langle A, N_a \rangle\}_{\text{pub}(B)} \\ B \rightarrow A : \quad \{\langle N_a, N_b \rangle\}_{\text{pub}(A)} \\ A \rightarrow B : \quad \quad \quad \{N_b\}_{\text{pub}(B)} \end{array}$$


## Questions

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

# Needham-Schroeder's Protocol (1978)


$$\begin{array}{l} A \rightarrow B : \quad \{\langle A, N_a \rangle\}_{\text{pub}(B)} \\ B \rightarrow A : \quad \{\langle N_a, N_b \rangle\}_{\text{pub}(A)} \\ A \rightarrow B : \quad \quad \quad \{N_b\}_{\text{pub}(B)} \end{array}$$


## Questions

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

## Attack

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

# Example : Man in the middle attack



Agent A



Intruder I



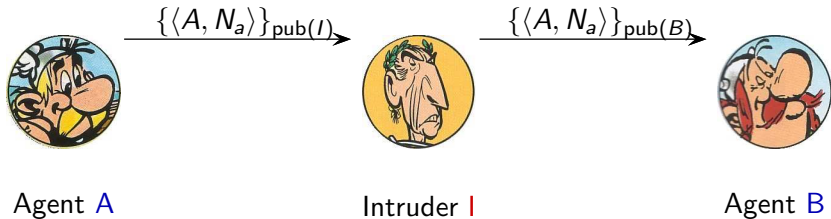
Agent B

## Attack

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

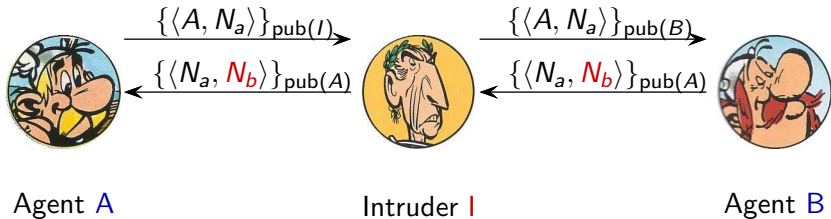
$$\begin{aligned} A \rightarrow B & : \{ \langle A, N_a \rangle \}_{\text{pub}(B)} \\ B \rightarrow A & : \{ \langle N_a, N_b \rangle \}_{\text{pub}(A)} \\ A \rightarrow B & : \{ N_b \}_{\text{pub}(B)} \end{aligned}$$

# Example : Man in the middle attack



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

# Example : Man in the middle attack



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





# Secrecy problem via constraint solving

→ for a fixed number of sessions

Protocol rules

in( $u_1$ ); out( $v_1$ )

in( $u_2$ ); out( $v_2$ )

...

in( $u_n$ ); out( $v_n$ )

Constraint System

$$\mathcal{C} = \left\{ \begin{array}{l} T_0 \stackrel{?}{\vdash} u_1 \\ T_0, v_1 \stackrel{?}{\vdash} u_2 \\ \dots \\ T_0, v_1, \dots, v_n \stackrel{?}{\vdash} s \end{array} \right.$$

# Secrecy problem via constraint solving

→ for a fixed number of sessions

Protocol rules

$\text{in}(u_1); \text{out}(v_1)$

$\text{in}(u_2); \text{out}(v_2)$

...

$\text{in}(u_n); \text{out}(v_n)$

Constraint System

$$\mathcal{C} = \left\{ \begin{array}{l} T_0 \vdash^? u_1 \\ T_0, v_1 \vdash^? u_2 \\ \dots \\ T_0, v_1, \dots, v_n \vdash^? s \end{array} \right.$$

## Solution of a constraint system in $\mathcal{I}$

A substitution  $\sigma$  such that

for every  $T \vdash^? u \in \mathcal{C}$ ,  $u\sigma$  is deducible from  $T\sigma$  in  $\mathcal{I}$ .

## Example : Needham-Schroeder's protocol

$A(a, I)$  and  $B(b)$  (running in parallel)

$$\begin{array}{l} \text{in}(\{\langle n_a, x_{n_b} \rangle\}_{\text{pub}(a)}) \quad ; \quad \text{out}(\{\langle a, n_a \rangle\}_{\text{pub}(I)}) \\ \text{in}(\{\langle y_a, y_{n_a} \rangle\}_{\text{pub}(b)}) \quad ; \quad \text{out}(\{\langle y_{n_a}, n_b \rangle\}_{\text{pub}(y_a)}) \end{array}$$

# Example : Needham-Schroeder's protocol

$A(a, I)$  and  $B(b)$  (running in parallel)

<b>1</b>		$\text{out}(\{\langle a, n_a \rangle\}_{\text{pub}(I)})$
<b>3</b>	$\text{in}(\{\langle n_a, x_{n_b} \rangle\}_{\text{pub}(a)})$	; $\text{out}(\{x_{n_b}\}_{\text{pub}(I)})$
<b>2</b>	$\text{in}(\{\langle y_a, y_{n_a} \rangle\}_{\text{pub}(b)})$	; $\text{out}(\{\langle y_{n_a}, n_b \rangle\}_{\text{pub}(y_a)})$

# Example : Needham-Schroeder's protocol

$A(a, I)$  and  $B(b)$  (running in parallel)

<b>1</b>		$\text{out}(\{\langle a, n_a \rangle\}_{\text{pub}(I)})$
<b>3</b>	$\text{in}(\{\langle n_a, x_{n_b} \rangle\}_{\text{pub}(a)})$	; $\text{out}(\{x_{n_b}\}_{\text{pub}(I)})$
<b>2</b>	$\text{in}(\{\langle y_a, y_{n_a} \rangle\}_{\text{pub}(b)})$	; $\text{out}(\{\langle y_{n_a}, n_b \rangle\}_{\text{pub}(y_a)})$

Constraints System

# Example : Needham-Schroeder's protocol

$A(a, I)$  and  $B(b)$  (running in parallel)

<b>1</b>		$\text{out}(\{\langle a, n_a \rangle\}_{\text{pub}(I)})$
<b>3</b>	$\text{in}(\{\langle n_a, x_{n_b} \rangle\}_{\text{pub}(a)})$	; $\text{out}(\{x_{n_b}\}_{\text{pub}(I)})$
<b>2</b>	$\text{in}(\{\langle y_a, y_{n_a} \rangle\}_{\text{pub}(b)})$	; $\text{out}(\{\langle y_{n_a}, n_b \rangle\}_{\text{pub}(y_a)})$

Constraints System

$T_0, \{a, n_a\}_{\text{pub}(I)}$

# Example : Needham-Schroeder's protocol

$A(a, I)$  and  $B(b)$  (running in parallel)

<b>1</b>		$\text{out}(\{\langle a, n_a \rangle\}_{\text{pub}(I)})$
<b>3</b>	$\text{in}(\{\langle n_a, x_{n_b} \rangle\}_{\text{pub}(a)})$	; $\text{out}(\{x_{n_b}\}_{\text{pub}(I)})$
<b>2</b>	$\text{in}(\{\langle y_a, y_{n_a} \rangle\}_{\text{pub}(b)})$	; $\text{out}(\{\langle y_{n_a}, n_b \rangle\}_{\text{pub}(y_a)})$

Constraints System

$$T_0, \{a, n_a\}_{\text{pub}(I)} \stackrel{?}{\vdash} \{y_a, y_{n_a}\}_{\text{pub}(b)}$$

# Example : Needham-Schroeder's protocol

$A(a, I)$  and  $B(b)$  (running in parallel)

**1** out( $\{\langle a, n_a \rangle\}_{\text{pub}(I)}$ )

**3** in( $\{\langle n_a, x_{n_b} \rangle\}_{\text{pub}(a)}$ ) ; out( $\{x_{n_b}\}_{\text{pub}(I)}$ )

**2** in( $\{\langle y_a, y_{n_a} \rangle\}_{\text{pub}(b)}$ ) ; out( $\{\langle y_{n_a}, n_b \rangle\}_{\text{pub}(y_a)}$ )

Constraints System

$T_0, \{a, n_a\}_{\text{pub}(I)} \stackrel{?}{\vdash} \{y_a, y_{n_a}\}_{\text{pub}(b)}$

$T_0, \{a, n_a\}_{\text{pub}(I)}, \{y_{n_a}, n_b\}_{\text{pub}(y_a)}$

# Example : Needham-Schroeder's protocol

$A(a, I)$  and  $B(b)$  (running in parallel)

**1** out( $\{\langle a, n_a \rangle\}_{\text{pub}(I)}$ )

**3** in( $\{\langle n_a, x_{n_b} \rangle\}_{\text{pub}(a)}$ ) ; out( $\{x_{n_b}\}_{\text{pub}(I)}$ )

**2** in( $\{\langle y_a, y_{n_a} \rangle\}_{\text{pub}(b)}$ ) ; out( $\{\langle y_{n_a}, n_b \rangle\}_{\text{pub}(y_a)}$ )

Constraints System

$$T_0, \{a, n_a\}_{\text{pub}(I)} \stackrel{?}{\vdash} \{y_a, y_{n_a}\}_{\text{pub}(b)}$$

$$T_0, \{a, n_a\}_{\text{pub}(I)}, \{y_{n_a}, n_b\}_{\text{pub}(y_a)} \stackrel{?}{\vdash} \{n_a, x_{n_b}\}_{\text{pub}(a)}$$

# Example : Needham-Schroeder's protocol

$A(a, I)$  and  $B(b)$  (running in parallel)

**1** out( $\{\langle a, n_a \rangle\}_{\text{pub}(I)}$ )

**3** in( $\{\langle n_a, x_{n_b} \rangle\}_{\text{pub}(a)}$ ) ; out( $\{x_{n_b}\}_{\text{pub}(I)}$ )

**2** in( $\{\langle y_a, y_{n_a} \rangle\}_{\text{pub}(b)}$ ) ; out( $\{\langle y_{n_a}, n_b \rangle\}_{\text{pub}(y_a)}$ )

Constraints System

$$T_0, \{a, n_a\}_{\text{pub}(I)} \stackrel{?}{\vdash} \{y_a, y_{n_a}\}_{\text{pub}(b)}$$

$$T_0, \{a, n_a\}_{\text{pub}(I)}, \{y_{n_a}, n_b\}_{\text{pub}(y_a)} \stackrel{?}{\vdash} \{n_a, x_{n_b}\}_{\text{pub}(a)}$$

$$T_0, \{a, n_a\}_{\text{pub}(I)}, \{y_{n_a}, n_b\}_{\text{pub}(y_a)}, \{x_{n_b}\}_{\text{pub}(I)}$$

# Example : Needham-Schroeder's protocol

$A(a, I)$  and  $B(b)$  (running in parallel)

**1** out( $\{\langle a, n_a \rangle\}_{\text{pub}(I)}$ )

**3** in( $\{\langle n_a, x_{n_b} \rangle\}_{\text{pub}(a)}$ ) ; out( $\{x_{n_b}\}_{\text{pub}(I)}$ )

**2** in( $\{\langle y_a, y_{n_a} \rangle\}_{\text{pub}(b)}$ ) ; out( $\{\langle y_{n_a}, n_b \rangle\}_{\text{pub}(y_a)}$ )

Constraints System

$$T_0, \{a, n_a\}_{\text{pub}(I)} \stackrel{?}{\vdash} \{y_a, y_{n_a}\}_{\text{pub}(b)}$$

$$T_0, \{a, n_a\}_{\text{pub}(I)}, \{y_{n_a}, n_b\}_{\text{pub}(y_a)} \stackrel{?}{\vdash} \{n_a, x_{n_b}\}_{\text{pub}(a)}$$

$$T_0, \{a, n_a\}_{\text{pub}(I)}, \{y_{n_a}, n_b\}_{\text{pub}(y_a)}, \{x_{n_b}\}_{\text{pub}(I)} \stackrel{?}{\vdash} n_b$$

# Example : Needham-Schroeder's protocol

$A(a, I)$  and  $B(b)$  (running in parallel)

<b>1</b>		$\text{out}(\{\langle a, n_a \rangle\}_{\text{pub}(I)})$
<b>3</b>	$\text{in}(\{\langle n_a, x_{n_b} \rangle\}_{\text{pub}(a)})$	; $\text{out}(\{x_{n_b}\}_{\text{pub}(I)})$
<b>2</b>	$\text{in}(\{\langle y_a, y_{n_a} \rangle\}_{\text{pub}(b)})$	; $\text{out}(\{\langle y_{n_a}, n_b \rangle\}_{\text{pub}(y_a)})$

Constraints System

$$\begin{aligned} T_0, \{a, n_a\}_{\text{pub}(I)} &\stackrel{?}{\vdash} \{y_a, y_{n_a}\}_{\text{pub}(b)} \\ T_0, \{a, n_a\}_{\text{pub}(I)}, \{y_{n_a}, n_b\}_{\text{pub}(y_a)} &\stackrel{?}{\vdash} \{n_a, x_{n_b}\}_{\text{pub}(a)} \\ T_0, \{a, n_a\}_{\text{pub}(I)}, \{y_{n_a}, n_b\}_{\text{pub}(y_a)}, \{x_{n_b}\}_{\text{pub}(I)} &\stackrel{?}{\vdash} n_b \end{aligned}$$

**Solution**  $\sigma = \{y_a \mapsto a, y_{n_a} \mapsto n_a, x_{n_b} \mapsto n_b\}$

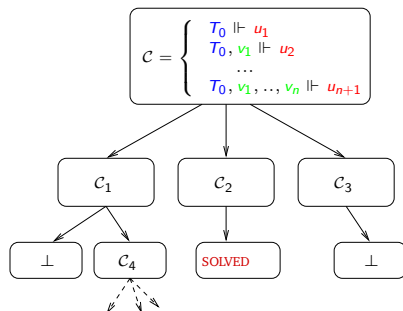
# Decision procedure

There exists an algorithm (actually a set of simplification rules) to decide whether such kind of constraint systems have a solution or not.

# Decision procedure

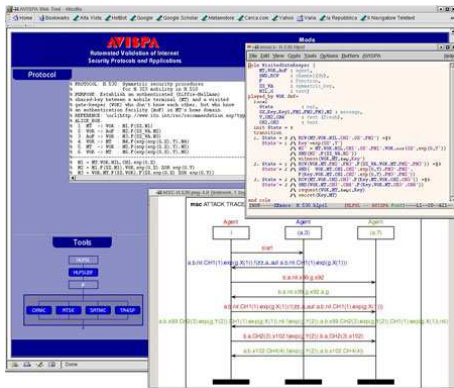
There exists an algorithm (actually a set of simplification rules) to decide whether such kind of constraint systems have a solution or not.

Main idea of the procedure :



# Outil de vérification AVISPA

Outil disponible en ligne : <http://www.avispa-project.org/>



→ Projet Européen (France, Italie, Allemagne, Suisse)

# Outline of the talk

- 1 Introduction
- 2 Cryptographic protocols
  - How protocols can be attacked ?
  - How protocols can be proved secure ?
- 3 Towards more realism
  - Algebraic properties
  - Composition
- 4 Conclusion

# Outline of the talk

## 1 Introduction

## 2 Cryptographic protocols

- How protocols can be attacked ?
- How protocols can be proved secure ?

## 3 Towards more realism

- Algebraic properties
- Composition

## 4 Conclusion

# Encryption modes (ECB/CBC)

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

Questions?

How can we encrypt long messages?

# Encryption modes (ECB/CBC)

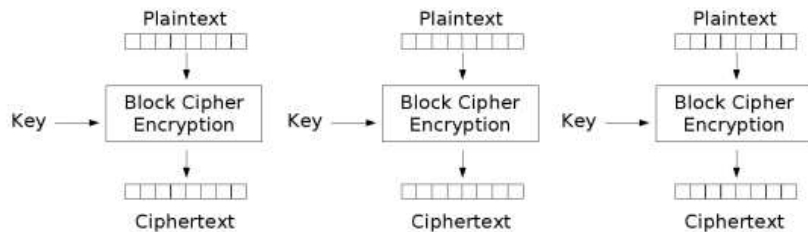
Encryption algorithms allow us to encrypt small messages (a block)

## Questions ?

How can we encrypt long messages ?

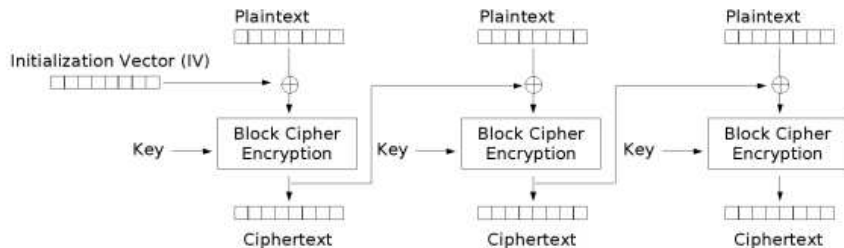
- **ECB** : Electronic CodeBook
- **CBC** : Cipher Block Chaining

# Electronic CodeBook



Electronic Codebook (ECB) mode encryption

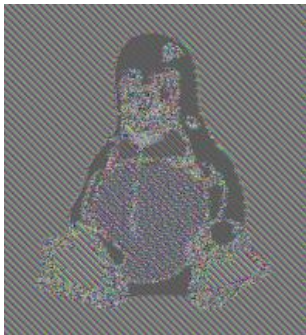
# Cipher Block Chaining



Cipher Block Chaining (CBC) mode encryption

# What about security ?

A pixel-map version of an image encrypted with



# Needham-Shroeder-Lowe protocol



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



# Needham-Shroeder-Lowe protocol



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



# Needham-Shroeder-Lowe protocol



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



# Needham-Shroeder-Lowe protocol



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



# Needham-Shroeder-Lowe protocol


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


## Questions

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

# Needham-Shroeder-Lowe protocol


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


## Questions

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

## Attacks ?

This protocol has been proved secure.

# El Gamal - a probabilistic encryption scheme

Consider a cyclic group  $G$  of order  $q$ , and a generator  $g$

Generating key pairs

private key :  $\text{priv}(a)$                       public key :  $\text{pub}(a) = g^{\text{priv}(a)}$

Encryption of  $m = g^{m'}$  with randomness  $r$

$$\{m\}_{\text{pub}(a)} \stackrel{\text{def}}{=} (\text{pub}(a)^r \times m, g^r) = (g^{(\text{priv}(a) \times r) + m'}, g^r)$$

Decryption of  $\{m\}_{\text{pub}(a)}$

- raising  $g^r$  to the power of  $\text{priv}(a)$ ;
- use this result for dividing the first component of the pair.

# El Gamal - a probabilistic encryption scheme

Consider a cyclic group  $G$  of order  $q$ , and a generator  $g$

Generating key pairs

private key :  $\text{priv}(a)$                       public key :  $\text{pub}(a) = g^{\text{priv}(a)}$

Encryption of  $m = g^{m'}$  with randomness  $r$

$$\{m\}_{\text{pub}(a)} \stackrel{\text{def}}{=} (\text{pub}(a)^r \times m, g^r) = (g^{(\text{priv}(a) \times r) + m'}, g^r)$$

Decryption of  $\{m\}_{\text{pub}(a)}$

- raising  $g^r$  to the power of  $\text{priv}(a)$ ;
- use this result for dividing the first component of the pair.

Sanity check

$$m, \text{pub}(a), r \xrightarrow{\text{encrypt}} \{m\}_{\text{pub}(a)}^r \xrightarrow{\text{decrypt}} m$$

# NSL protocol with El Gamal encryption

Pairing two group elements  $m_1 = g^{m'_1}$  and  $m_2 = g^{m'_2}$ .

$$\langle m_1, m_2 \rangle \stackrel{\text{def}}{=} g^{m'_1 + 2^{|m'_1|} \times m'_2}$$

→ we concatenate the binary representation of  $m'_1$  and  $m'_2$ .

## Attack !

The Lowe Fixed version of the Needham-Schroeder protocol becomes now insecure.

# Example : Man in the middle attack



Agent A

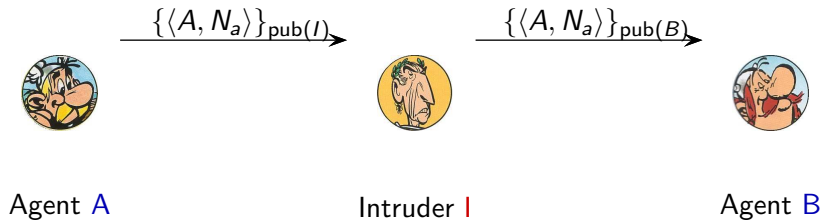


Intruder I

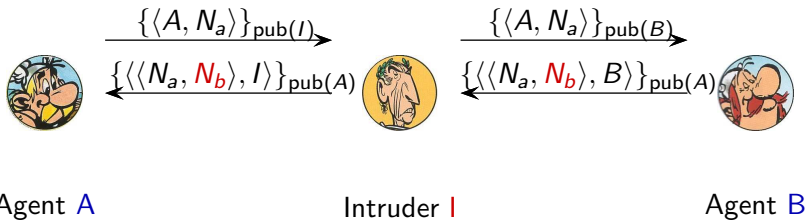


Agent B

# Example : Man in the middle attack



# Example : Man in the middle attack



# Example : Man in the middle attack



Agent A

Intruder I

Agent B

Let  $\alpha$  be the length of a nonce. We have that :

- $\{\langle \langle N_a, N_b \rangle, B \rangle\}_{\text{pub}(A)} = (g^{N_a+2^\alpha \times N_b+2^{2\alpha} \times B} \times \text{pub}(A)^r, g^r)$
- $\{\langle \langle N_a, N_b \rangle, I \rangle\}_{\text{pub}(A)} = (g^{N_a+2^\alpha \times N_b+2^{2\alpha} \times I} \times \text{pub}(A)^r, g^r)$

# Example : Man in the middle attack



Agent A

Intruder I

Agent B

Let  $\alpha$  be the length of a nonce. We have that :

- $\{\langle \langle N_a, N_b \rangle, B \rangle\}_{\text{pub}(A)} = (g^{N_a+2^\alpha \times N_b+2^{2\alpha} \times B} \times \text{pub}(A)^r, g^r)$
- $\{\langle \langle N_a, N_b \rangle, I \rangle\}_{\text{pub}(A)} = (g^{N_a+2^\alpha \times N_b+2^{2\alpha} \times I} \times \text{pub}(A)^r, g^r)$

How can the attacker obtain  $\{\langle \langle N_a, N_b \rangle, I \rangle\}_{\text{pub}(A)}$  ?

Multiplication with  $g^{-2^{2\alpha} \times B} \times g^{2^{2\alpha} \times I}$



# More algebraic properties

Some protocols used quite complex cryptographic primitives

# More algebraic properties

Some protocols used quite complex cryptographic primitives

**Example** : blind signature in e-voting

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

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

Then, the voter can obtain  $\text{sgn}(v, ska)$  by using  $r$ .

# More algebraic properties

Some protocols used quite complex cryptographic primitives

**Example** : blind signature in e-voting

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

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

Then, the voter can obtain  $\text{sgn}(v, ska)$  by using  $r$ .

Of course, this signature scheme can not be abstracted as a classical signature scheme.

# Outline of the talk

## 1 Introduction

## 2 Cryptographic protocols

- How protocols can be attacked ?
- How protocols can be proved secure ?

## 3 Towards more realism

- Algebraic properties
- **Composition**

## 4 Conclusion

## Formal verification of security protocols

- Existing tools allow us to verify **relatively small** protocols and sometimes only for a **bounded number of sessions**
- Most often, we verify them in **isolation**  
→ this is not sufficient

## Formal verification of security protocols

- Existing tools allow us to verify **relatively small** protocols and sometimes only for a **bounded number of sessions**
- Most often, we verify them in **isolation**  
→ this is not sufficient

### Example :

$$P_1 : A \rightarrow B : \{s\}_{\text{pub}(B)}$$

Question : What about the secrecy of  $s$ ?

# Motivations

## Formal verification of security protocols

- Existing tools allow us to verify **relatively small** protocols and sometimes only for a **bounded number of sessions**
- Most often, we verify them in **isolation**  
→ this is not sufficient

### Example :

$$P_1 : A \rightarrow B : \{s\}_{\text{pub}(B)}$$

$$P_2 : \begin{array}{l} A \rightarrow B : \{N_a\}_{\text{pub}(B)} \\ B \rightarrow A : N_a \end{array}$$

Question : What about the secrecy of  $s$ ?

## Formal verification of security protocols

- Existing tools allow us to verify **relatively small** protocols and sometimes only for a **bounded number of sessions**
- Most often, we verify them in **isolation**  
→ this is not sufficient

## Our goal

investigate **sufficient conditions** to ensure that protocols can be safely used in an environment where :

- ① other sessions of the **same protocol** may be executed ;
- ② other sessions of **another protocol** may be executed as well.

→ protocols may share identities and keys (e.g. public keys, long-term symmetric keys)

## Tagged protocols

Each protocol (or each session) is given an **identifier** (e.g. the protocol's name, or a session identifier). This identifier has to appear in any **encrypted** and **signed** message.

→ this **tagging policy** will avoid interaction between two different protocols or different sessions.

# Main condition - Tagging

## Tagged protocols

Each protocol (or each session) is given an **identifier** (e.g. the protocol's name, or a session identifier). This identifier has to appear in any **encrypted** and **signed** message.

→ this **tagging policy** will avoid interaction between two different protocols or different sessions.

**Example** :  $P_1$  is 1-tagged whereas  $P_2$  is 2-tagged

Protocol  $P_1$

$$A \rightarrow B : \{\langle 1, s \rangle\}_{\text{pub}(B)}$$

Protocol  $P_2$

$$A \rightarrow B : \{\langle 2, N_a \rangle\}_{\text{pub}(B)}$$

$$B \rightarrow A : N_a$$

## Theorem

Let  $P_1$  and  $P_2$  be two *well-tagged* protocols such that

- no critical long-term keys appear in plaintext position neither in  $P_1$  nor in  $P_2$ ,
- $P_1$  is  $\alpha$ -tagged and  $P_2$  is  $\beta$ -tagged with  $\alpha \neq \beta$ .

If  $P_1$  preserves the secrecy of  $s$  then  $P_1 \mid P_2$  preserves the secrecy of  $s$ .

## Theorem

Let  $P_1$  and  $P_2$  be two *well-tagged* protocols such that

- no critical long-term keys appear in plaintext position neither in  $P_1$  nor in  $P_2$ ,
- $P_1$  is  $\alpha$ -tagged and  $P_2$  is  $\beta$ -tagged with  $\alpha \neq \beta$ .

If  $P_1$  preserves the secrecy of  $s$  then  $P_1 \mid P_2$  preserves the secrecy of  $s$ .

Some extensions that have been already done :

- 1 well-tagged condition can be relaxed : **disjoint encryption** is actually sufficient ;
- 2 composition result holds for a **class of security properties** (secrecy, authentication, ...)
- 3 Other kinds of composition (*e.g.* sequential composition)

# Composition : “from one session to many”

Our goal :

compose **different sessions** from the **same** protocol

**Main difficulty** : The participants need to agree on a common session identifier before starting the protocol.

→ A static tag, as the name of the protocol, is not sufficient. Why?

# Composition : “from one session to many”

Our goal :

compose **different sessions** from the **same** protocol

**Main difficulty** : The participants need to agree on a common session identifier before starting the protocol.

→ A static tag, as the name of the protocol, is not sufficient. Why?

Solution

- a **transformation** which maps a protocol  $P$  that is secure for a **single session** to a protocol  $\overline{P}$  that is secure for an **unbounded number of sessions**.
- **side-effect** : an effective strategy to design secure protocols

# Transformation

Let  $P$  be a protocol with  $\ell$  participants as given below :

$$A_{i_1} \rightarrow A_{j_1} : m_1$$

$$A_{i_2} \rightarrow A_{j_2} : m_2$$

$$\vdots$$

$$A_{i_k} \rightarrow A_{j_k} : m_k$$

# Transformation

The protocol  $\overline{P}$  (with  $\ell$  participants) is described below :

**Initialisation phase** : broadcast of fresh nonces

$$A_1 \rightarrow All : A_1, N_1$$

$$A_2 \rightarrow All : A_2, N_2$$

$\vdots$

$$A_\ell \rightarrow All : A_\ell, N_\ell$$

# Transformation

The protocol  $\overline{P}$  (with  $\ell$  participants) is described below :

**Initialisation phase** : broadcast of fresh nonces

$$A_1 \rightarrow All : A_1, N_1$$

$$A_2 \rightarrow All : A_2, N_2$$

$\vdots$

$$A_\ell \rightarrow All : A_\ell, N_\ell$$

Every participant obtain a **tag** =  $\langle A_1, N_1, A_2, N_2, \dots, A_\ell, N_\ell \rangle$

# Transformation

The protocol  $\overline{P}$  (with  $\ell$  participants) is described below :

**Initialisation phase** : broadcast of fresh nonces

$$A_1 \rightarrow All : A_1, N_1$$

$$A_2 \rightarrow All : A_2, N_2$$

$\vdots$

$$A_\ell \rightarrow All : A_\ell, N_\ell$$

Every participant obtain a **tag** =  $\langle A_1, N_1, A_2, N_2, \dots, A_\ell, N_\ell \rangle$

**Main phase** :

where the function  $\overline{m}$  is defined by :

$$A_{i_1} \rightarrow A_{j_1} : \overline{m_1}$$

$$A_{i_2} \rightarrow A_{j_2} : \overline{m_2}$$

$\vdots$

$$A_{i_k} \rightarrow A_{j_k} : \overline{m_k}$$

$$\left\{ \begin{array}{ll} \overline{\langle u_1, u_2 \rangle} & \rightarrow \langle \overline{u_1}, \overline{u_2} \rangle \\ \overline{f(u_1, u_2)} & \rightarrow f(\langle \text{tag}, \overline{u_1} \rangle, \overline{u_2}) \\ & \text{when } f \in \{\text{enc}, \text{aenc}, \text{sign}\} \\ \overline{u} & \rightarrow u \quad \text{otherwise} \end{array} \right.$$

# Composition result “from one session to many”

## Theorem

Let  $P$  be a protocol with no **critical long-term keys** in plaintext position.

If  $P$  preserves the secrecy of  $s$  for a **single honest session** of each role then  $\overline{P}$  preserves the secrecy of  $s$  for an **unbounded number of sessions**.

- **critical long-term keys** do not appear in plaintext
  - this can be easily checked on the finite specification of the protocol
  - often satisfied since it is considered as a prudent practice
- **single honest session** of each role
  - i.e. one an instance of each role (in general 2 or 3);
  - participants engaged in this session are honest.

# Composition result “from one session to many”

## Theorem

Let  $P$  be a protocol with no **critical long-term keys** in plaintext position.

If  $P$  preserves the secrecy of  $s$  for a **single honest session** of each role then  $\overline{P}$  preserves the secrecy of  $s$  for an **unbounded number of sessions**.

- **critical long-term keys** do not appear in plaintext
  - this can be easily checked on the finite specification of the protocol
  - often satisfied since it is considered as a prudent practice
- **single honest session** of each role
  - i.e. one an instance of each role (in general 2 or 3);
  - participants engaged in this session are honest.

**Exemple** : Needham-Schroeder public key protocol

→ the Lowe’s famous man-in-the-middle attack is prevented

# Outline of the talk

- 1 Introduction
- 2 Cryptographic protocols
  - How protocols can be attacked ?
  - How protocols can be proved secure ?
- 3 Towards more realism
  - Algebraic properties
  - Composition
- 4 Conclusion

How can we verify cryptographic protocols ?

- modelling the protocol, the security properties
- manually / automatically
  - the problem is **undecidable** in general (some tools exist)

It remains a lot to do

- modelling security properties is a **difficult task**
- take into account the **algebraic properties** of the primitives
- developing methods and tools to check equivalence-based security properties (e.g. **privacy** properties)
- analyse the **source code** of the protocol instead of its specification
- ...