

Providing solutions for more secure exchanges

Stéphanie Delaune

November 18, 2014



Cryptographic protocols

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



Cryptographic protocols

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

The network is unsecure!

Communications take place over a **public** network like the Internet.

Cryptographic protocols

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



Security properties

- **Secrecy**: May an intruder learn some secret message between two honest participants?
- **Authentication**: Is the agent **Alice** really talking to **Bob**?
- **Fairness**: **Alice** and **Bob** want to sign a contract. **Alice** initiates the protocol. May **Bob** obtain some advantage?
- **Privacy**: **Alice** participate to an election. May a participant learn something about the vote of **Alice**?
- **Non-repudiation**: **Alice** sends a message to **Bob**. **Alice** cannot later deny having sent this message. **Bob** cannot deny having received the message.
- ...

Cryptographic primitives

Cryptographic primitives

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

Cryptographic primitives

Cryptographic primitives

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

Symmetric encryption



Cryptographic primitives

Cryptographic primitives

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

Asymmetric encryption



Symmetric vs. asymmetric encryption

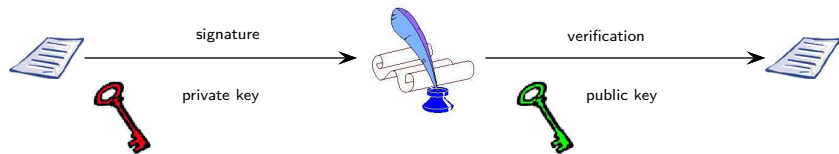
Symmetric encryption

- **efficient** in practice,
- agents have to **share a secret key**
 - trusted third party, distribution key protocol

Asymmetric encryption

- **not efficient** in practice,
- agents do not have to share a secret
 - often used in establishment key protocols
- **authentication** of public keys (certificate)

Digital signature: How does it work?



- similar to public key encryption
- everyone knows the key to verify the signature (**public** key)
- the key used to sign a message has to be **private** (**private** key)

Properties

- the signature has to **authenticate** the signer
- the signature “ belongs to “ **one** particular **document**
- the signed document **can not be modified** afterwards

Properties and applications

Properties

- the signature has to **authenticate** the signer
- the signature “ belongs to “ **one** particular **document**
- the signed document **can not be modified** afterwards

Applications

- certificate to authenticate a public key
- contract signing protocols
- E-voting protocols (blind signature)

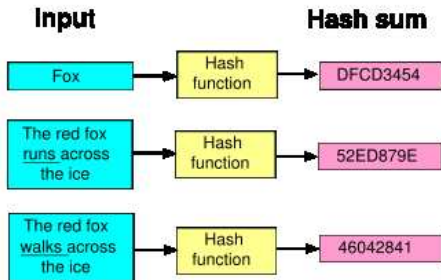
→ allows someone to sign without knowing the message he is signing.

Hash function: What is it?

Hash function

It is a reproducible method of turning some kind of data into a (relatively) **small number** that may serve as a digital "**fingerprint**" of the data (again substitutions and permutations).

Examples: MD5, SHA-1



Properties and Applications

Properties

- **deterministic** function
- **one-way function**: there is no practical way to retrieve m from $hash(m)$
- **collision resistant**: difficult to find m_1 and m_2 such that $m_1 \neq m_2$ and $hash(m_1) = hash(m_2)$

Some applications

- to improve **efficiency**: we can sign $hash(m)$ instead of m
- use to **guarantee the integrity** of a message
- checksum to detect errors

Outline of the talk

- 1 Introduction
- 2 Some examples of security protocols
 - Credit Card payment
 - Needham Schroeder protocol
- 3 How can we verify them?
 - How protocols can be attacked?
 - How protocols can be proved secure?
- 4 Conclusion

Outline of the talk

- 1 Introduction
- 2 Some examples of security protocols
 - Credit Card payment
 - Needham Schroeder protocol
- 3 How can we verify them?
 - How protocols can be attacked?
 - How protocols can be proved secure?
- 4 Conclusion

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.

Example: credit card payment



- The client C puts his credit card C in the terminal T .
- The merchant enters the amount M of the sale.
- The terminal authenticates the credit card.
- The client enters his PIN.
If $M \geq 100\text{€}$, then in 20% of cases,
 - The terminal contacts the bank B .
 - The banks gives its authorisation.



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

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

Bank

- a **private** signature key – $\text{priv}(B)$
- a **public** key to verify a signature – $\text{pub}(B)$
- a **secret** key shared with the credit card – K_{CB}

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

Bank

- a **private** signature key – $\text{priv}(B)$
- a **public** key to verify a signature – $\text{pub}(B)$
- a **secret** key shared with the credit card – K_{CB}

Credit Card

- some **Data**: name of the cardholder, expiry date ...
- a signature of the **Data** – $\{\text{hash}(\text{Data})\}_{\text{priv}(B)}$
- a **secret** key shared with the bank – K_{CB}

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

Bank

- a **private** signature key – $\text{priv}(B)$
- a **public** key to verify a signature – $\text{pub}(B)$
- a **secret** key shared with the credit card – K_{CB}

Credit Card

- some **Data**: name of the cardholder, expiry date ...
- a signature of the **Data** – $\{\text{hash}(\text{Data})\}_{\text{priv}(B)}$
- a **secret** key shared with the bank – K_{CB}

Terminal

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

Payment protocol

the terminal T reads the credit card C :

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

Payment protocol

the terminal T reads the credit card C :

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

the terminal T asks the code:

2. $T \rightarrow CI : code?$

3. $CI \rightarrow C : 1234$

4. $C \rightarrow T : ok$

Payment protocol

the terminal T reads the credit card C :

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

the terminal T asks the code:

2. $T \rightarrow CI$: $code?$

3. $CI \rightarrow C$: 1234

4. $C \rightarrow T$: ok

the terminal T requests authorisation the bank B :

5. $T \rightarrow B$: $auth?$

6. $B \rightarrow T$: 4528965874123

7. $T \rightarrow C$: 4528965874123

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

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

10. $B \rightarrow T$: ok

Faible sur la carte bleue

Initialement la sécurité été assurée par :

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



Faible sur la carte bleue

Initialement la sécurité été assurée par :

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



Mais il y a des failles !

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



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

La « YesCard »: Comment ça marche ?

Faible logique

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

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

3. $CI \rightarrow C$: 1234

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

La « YesCard » : Comment ça marche ?

Faible logique

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

La « YesCard » : Comment ça marche ?

Faible logique

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

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

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

La « YesCard » : Comment ça marche ?

Faible logique

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

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

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

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

Outline of the talk

- 1 Introduction
- 2 Some examples of security protocols
 - Credit Card payment
 - Needham Schroeder protocol
- 3 How can we verify them?
 - How protocols can be attacked?
 - How protocols can be proved secure?
- 4 Conclusion

Needham-Schroeder's Protocol (1978)



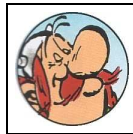
- $A \rightarrow B : \{\langle A, N_a \rangle\}_{\text{pub}(B)}$
 $B \rightarrow A : \{\langle N_a, N_b \rangle\}_{\text{pub}(A)}$
 $A \rightarrow B : \{N_b\}_{\text{pub}(B)}$



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)



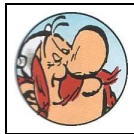
$A \rightarrow B : \{\langle A, N_a \rangle\}_{\text{pub}(B)}$
 $B \rightarrow A : \{\langle N_a, N_b \rangle\}_{\text{pub}(A)}$
• $A \rightarrow B : \{N_b\}_{\text{pub}(B)}$



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)


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


Questions

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

Needham-Schroeder's Protocol (1978)



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



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



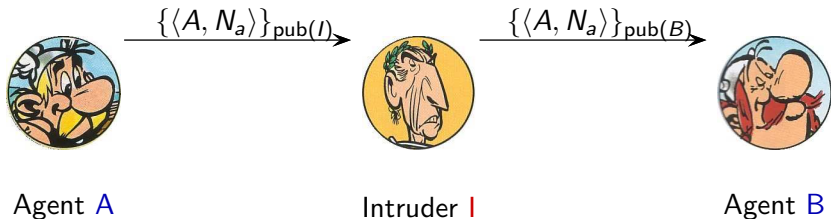
Agent B

Attack

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

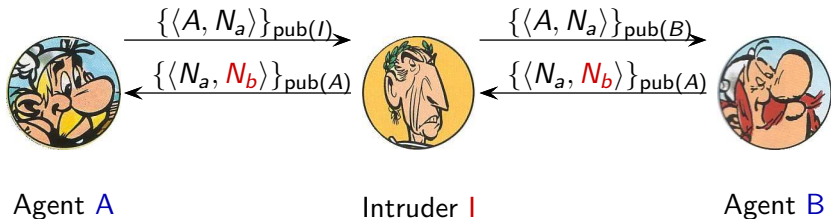
$$\begin{aligned} A \rightarrow B & : \{ \langle A, N_a \rangle \}_{\text{pub}(B)} \\ B \rightarrow A & : \{ \langle N_a, N_b \rangle \}_{\text{pub}(A)} \\ A \rightarrow B & : \{ N_b \}_{\text{pub}(B)} \end{aligned}$$

Example: Man in the middle attack



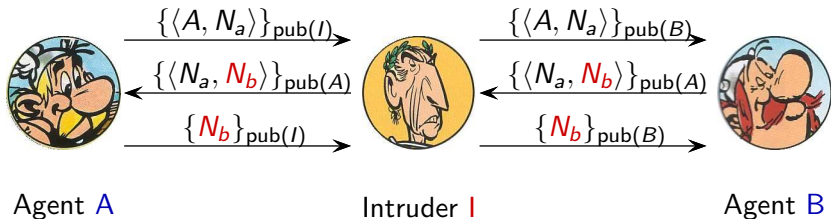
- $A \rightarrow B : \{\langle A, N_a \rangle\}_{\text{pub}(B)}$
- $B \rightarrow A : \{\langle N_a, N_b \rangle\}_{\text{pub}(A)}$
- $A \rightarrow B : \{N_b\}_{\text{pub}(B)}$

Example: Man in the middle attack



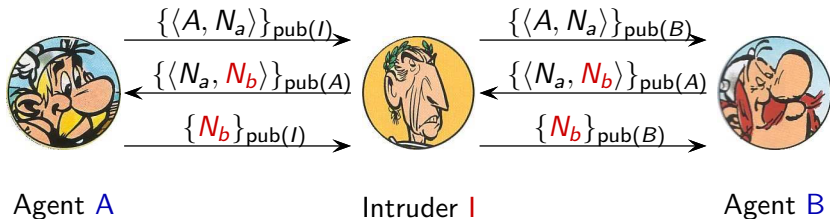
$A \rightarrow B : \{\langle A, N_a \rangle\}_{\text{pub}(B)}$
 $B \rightarrow A : \{\langle N_a, N_b \rangle\}_{\text{pub}(A)}$
 $A \rightarrow B : \{N_b\}_{\text{pub}(B)}$

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



Attack

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

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

Outline of the talk

- 1 Introduction
- 2 Some examples of security protocols
 - Credit Card payment
 - Needham Schroeder protocol
- 3 How can we verify them?
 - How protocols can be attacked?
 - How protocols can be proved secure?
- 4 Conclusion

—→ 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

→ 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 ?

∞ ∞ ∞

Comment fait-on ?



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”

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 Some examples of security protocols
 - Credit Card payment
 - Needham Schroeder protocol
- 3 How can we verify them?
 - How protocols can be attacked?
 - How protocols can be proved secure?
- 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.

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

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	

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?

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

Examples:

- **man in the middle** attacks: *e.g.* Needham Schroeder protocol;
- **replay** attacks: electronic passport protocol (French version),
electronic voting protocol (*e.g.* Helios).

Outline of the talk

- 1 Introduction
- 2 Some examples of security protocols
 - Credit Card payment
 - Needham Schroeder protocol
- 3 How can we verify them?
 - How protocols can be attacked?
 - How protocols can be proved secure?
- 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

Symbolic models

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](#)]

Symbolic model

Messages are abstracted by terms

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

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 $\text{senc}(m, k)$ and public key encryption $\text{aenc}(m, \text{pub}(A))$,
- signature $\text{sign}(m, \text{priv}(A))$.

Presence of an idealized attacker

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



Examples:

$$\frac{m \quad k}{\text{senc}(m, k)} \qquad \frac{\text{senc}(m, k) \quad k}{m} \qquad \frac{\text{aenc}(m, \text{pub}(a)) \quad \text{priv}(a)}{m}$$

Presence of an attacker ...



Presence of an attacker ...

who controls the communication network:

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



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

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**.

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**.

Secrecy problem in presence of a passive attacker

Intruder deduction problem for a fixed inference system \mathcal{I}

Input: a finite set of ground terms T (the knowledge of the attacker) and a ground term s (the secret),

Output: Is s deducible from T in \mathcal{I} ?

Secrecy problem in presence of a passive attacker

Intruder deduction problem for a fixed inference system \mathcal{I}

Input: a finite set of ground terms T (the knowledge of the attacker) and a ground term s (the secret),

Output: Is s deducible from T in \mathcal{I} ?

Example: $T = \{\text{senc}(s_1, k_1); \text{senc}(s_2, k_2); \langle k_1, k_2 \rangle\}$ and $s = \langle s_1, s_2 \rangle$.

$$\frac{x \quad y}{\langle x, y \rangle} \quad \frac{\langle x, y \rangle}{x} \quad \frac{\langle x, y \rangle}{y} \quad \frac{x \quad y}{\text{senc}(x, y)} \quad \frac{\text{senc}(x, y) \quad y}{x}$$

Secrecy problem in presence of a passive attacker

Intruder deduction problem for a fixed inference system \mathcal{I}

Input: a finite set of ground terms T (the knowledge of the attacker) and a ground term s (the secret),

Output: Is s deducible from T in \mathcal{I} ?

Example: $T = \{\text{senc}(s_1, k_1); \text{senc}(s_2, k_2); \langle k_1, k_2 \rangle\}$ and $s = \langle s_1, s_2 \rangle$.

$$\frac{x \quad y}{\langle x, y \rangle} \quad \frac{\langle x, y \rangle}{x} \quad \frac{\langle x, y \rangle}{y} \quad \frac{x \quad y}{\text{senc}(x, y)} \quad \frac{\text{senc}(x, y) \quad y}{x}$$

Results

The intruder deduction problem is decidable in PTIME for the inference system given above (and some others)

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

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

Solution of a constraint system in \mathcal{I}

A substitution σ such that

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

Example: Needham-Schroeder's protocol

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

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

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

Example: Needham-Schroeder's protocol

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

- 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($\{\langle y_{n_a}, n_b \rangle\}_{\text{pub}(y_a)}$)

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

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

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

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

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

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

Constraints System

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

Example: Needham-Schroeder's protocol

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

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

Constraints System

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

Example: Needham-Schroeder's protocol

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

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

Constraints System

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

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

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

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

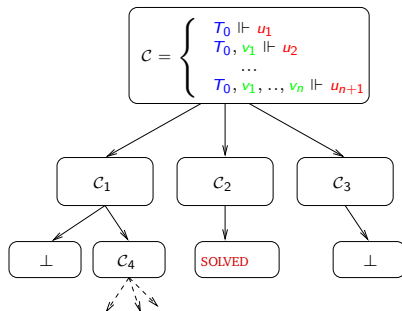
Decision procedure

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

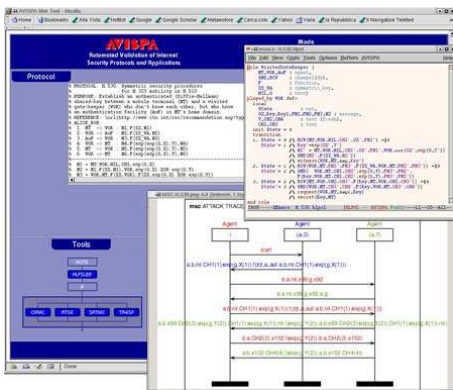
There exists an algorithm (actually a set of simplification rules) to decide whether such kind of constraint systems have a solution or not.

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 Some examples of security protocols
 - Credit Card payment
 - Needham Schroeder protocol
- 3 How can we verify them?
 - How protocols can be attacked?
 - How protocols can be proved secure?
- 4 Conclusion

Conclusion

Cryptographic protocols

- numerous, various security goals
- can be **attacked** even if the primitives are **secure**
→ <http://www.lsv.ens-cachan.fr/spore/>

How to verify them?

- **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**
- does a suitable E-voting protocol exist?
- take into account the **algebraic properties** of the primitives
- analyse the source code of the protocol instead of its specification