


Verification of Security Protocols

Véronique Cortier¹

July 2nd, 2010

Movep 2010

¹LORIA, CNRS - INRIA Cassis project, Nancy Universities 

LORIA (Nancy)



Size : 500 researchers, among which about 150 permanent researchers and 150 PhD students.

Where is it ?



Well connected to :

- Paris, France (90 minutes)
- Luxembourg (90-120 minutes)
- Saarbrücken, Germany (120 minutes)

What kind of research ?

Research themes

- High-performance calculations, simulation and visualization
- Model checking, security, rewriting systems
- Parallel, distributed and communicating systems
- Models and algorithms for bio-sciences
- Natural Language Processing and multi-modal communication
- Knowledge representation and processing

Regular job offers !

- PhD positions
- Post-doc positions
- Permanent positions (CNRS, INRIA, Universities)

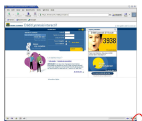
Outline of the talk

- 1 Introduction on security protocols
 - Context
 - Security Protocols : how does it work ?
 - Commutative encryption (RSA)
 - Needham-Schroeder Example
- 2 Formal models
 - Messages
 - Intruder
 - Protocol
 - Solving constraint systems
- 3 Going further
 - Undecidability
 - Horn clauses
 - Adding equational theories
 - Some results
- 4 Towards more guarantees
 - Cryptographic models
 - Linking Formal and cryptographic models
 - Extension to indistinguishability

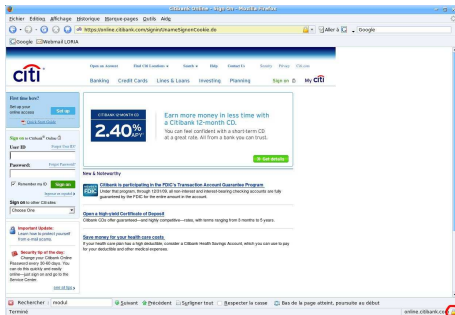
Context : cryptographic protocols

Cryptographic protocols are widely used in everyday life.

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



On the web



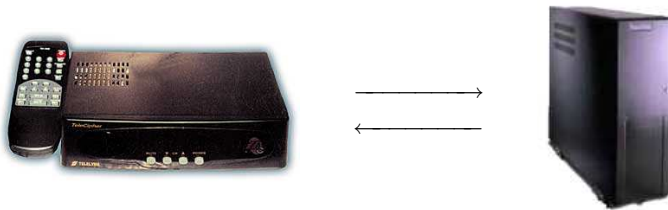
- HTTPS, i.e. the SSL protocol for ensuring confidentiality
- password-based authentication

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



- The result corresponds to the votes.
- Each vote is confidential.
- No partial result is leaked before the end of the election
- Only voters can vote and at most once
- Coercion resistance

Electronic purse



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

Security goals

Cryptographic protocols aim at

- preserving confidentiality of data
(e.g. pin code, medical files, ...)
- ensuring authenticity
(Are you really talking to your bank ??)
- ensuring anonymous communications
(for e-voting protocols, ...)
- protecting against repudiation
(I never sent this message !!)
- ...

⇒ Cryptographic protocols vary depending on the application.

How does this work ?

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, ...)

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 : **B**ank, **C**ustomer, **C**ard and **T**erminal.

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 :

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

Credit card payment Protocol (in short)

The terminal reads the card :

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

The terminal asks for the secret code :

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

$$3. \quad C_u \rightarrow C_a : 1234$$

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

Credit card payment Protocol (in short)

The terminal reads the card :

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

The terminal asks for the secret code :

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

$$3. \quad C_u \rightarrow C_a : 1234$$

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

The terminal calls the bank :

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

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

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

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

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

$$10. \quad B \rightarrow T : \text{ok}$$

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 flow

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 flow

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*

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

How does the “YesCard” work ?

Logical flaw

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

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

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.

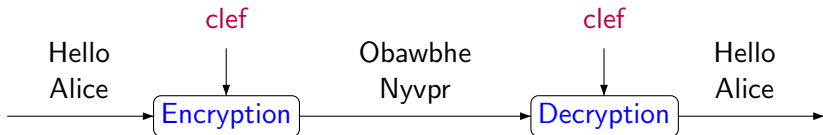


Challenge

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)

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

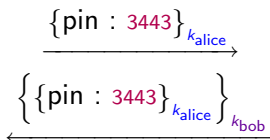
Exchanging a secret with commutative encryption (RSA)



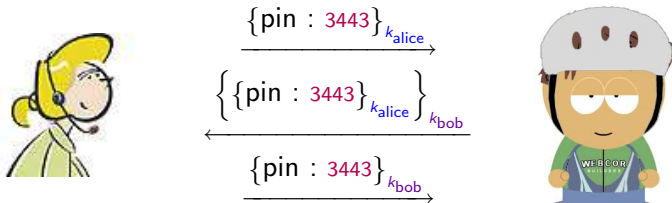
$\{\text{pin} : 3443\}_{k_{\text{alice}}}$
→



Exchanging a secret with commutative encryption (RSA)

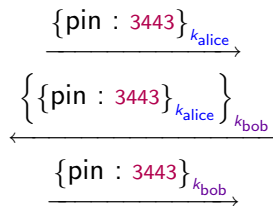


Exchanging a secret with commutative encryption (RSA)



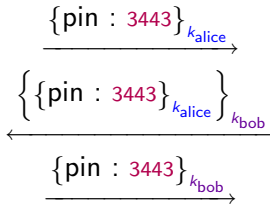
$$\text{Since } \left\{ \left\{ \text{pin} : 3443 \right\}_{k_{\text{alice}}} \right\}_{k_{\text{bob}}} = \left\{ \left\{ \text{pin} : 3443 \right\}_{k_{\text{bob}}} \right\}_{k_{\text{alice}}}$$

Exchanging a secret with commutative encryption (RSA)

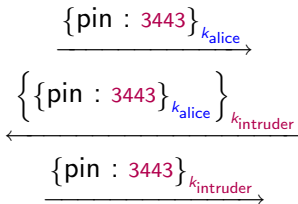


→ It does not work! (Authentication problem)

Exchanging a secret with commutative encryption (RSA)



→ It does not work! (Authentication problem)



Another example

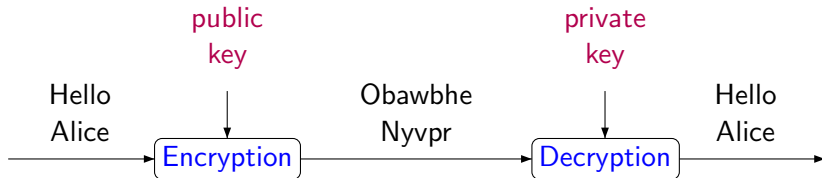
The “famous” Needham-Schroeder public key protocol

(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 .


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


Questions :

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

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

Man in the middle attack



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



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



Man in the middle attack



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



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



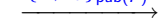
$\{N_a, N_b\}_{\text{pub}(A)}$

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

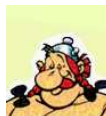
Man in the middle attack



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



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



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



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



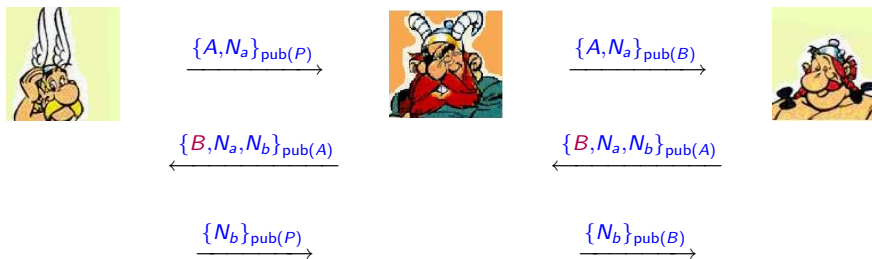
$\{N_b\}_{\text{pub}(P)}$



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



Man in the middle attack



Fixing the flaw : add the identity of B .

Outline of the talk

- 1 Introduction on security protocols
 - Context
 - Security Protocols : how does it work ?
 - Commutative encryption (RSA)
 - Needham-Schroeder Example
- 2 Formal models
 - Messages
 - Intruder
 - Protocol
 - Solving constraint systems
- 3 Going further
 - Undecidability
 - Horn clauses
 - Adding equational theories
 - Some results
- 4 Towards more guarantees
 - Cryptographic models
 - Linking Formal and cryptographic models
 - Extension to indistinguishability
 - Conclusion

Difficulty

Presence of an **attacker**

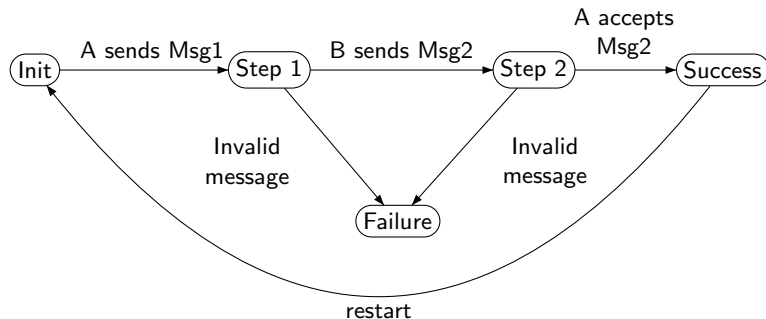
- may **read** every message sent on the net,
- may **intercept and send** new messages.



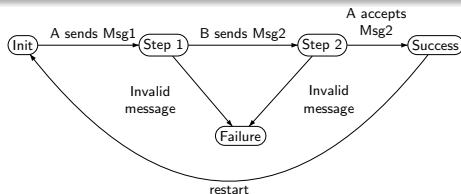
⇒ The system is infinitely branching

A naive approach

Why not modeling security protocol using a (possibly extended) automata ?



How to model a security protocol ?



- The output of each participants **strongly depends on the data received inside the message.**
- At each step, a malicious user (called the adversary) may **create arbitrary messages.**
- The output of the adversary **strongly depends on the messages sent on the network.**

→ It is important to have a tight modeling of the messages.

An appropriate datastructure : Terms

Given a **signature** \mathcal{F} of symbols with an arity

e.g. $\{\text{enc}, \text{pair}, a, b, c, n_a, n_b\}$

and a set \mathcal{X} of variables,

the set of **terms** $T(\mathcal{F}, \mathcal{X})$ is inductively defined as follows :

- constants terms (e.g. a, b, c, n_a, n_b) are terms
- variables are terms
- $f(t_1, \dots, t_n)$ is a term whenever t_1, \dots, t_n are terms.

Intuition : from words to trees.

→ There exists automata on trees instead of (classical) automata on words, see e.g. **TATA** <http://tata.gforge.inria.fr/>

Messages

Messages are abstracted by terms.

Agents : a, b, \dots

Nonces : n_1, n_2, \dots

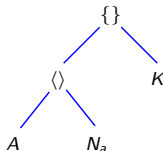
Keys : k_1, k_2, \dots

Cyphertext : $enc(m, k)$

Concatenation : $pair(m_1, m_2)$

Example : The message $\{A, N_a\}_K$ is represented by :

$enc(pair(A, N_a), K)$



Intuition : only the structure of the message is kept.

Intruder abilities

Composition rules

$$\frac{T \vdash u \quad T \vdash v}{T \vdash \langle u, v \rangle} \quad \frac{T \vdash u \quad T \vdash v}{T \vdash \text{enc}(u, v)} \quad \frac{T \vdash u \quad T \vdash v}{T \vdash \text{enca}(u, v)}$$



Intruder abilities

Composition rules

$$\frac{T \vdash u \quad T \vdash v}{T \vdash \langle u, v \rangle} \quad \frac{T \vdash u \quad T \vdash v}{T \vdash \text{enc}(u, v)} \quad \frac{T \vdash u \quad T \vdash v}{T \vdash \text{enca}(u, v)}$$



Decomposition rules

$$\frac{}{T \vdash u} u \in T \quad \frac{T \vdash \langle u, v \rangle}{T \vdash u} \quad \frac{T \vdash \langle u, v \rangle}{T \vdash v}$$

$$\frac{T \vdash \text{enc}(u, v) \quad T \vdash v}{T \vdash u} \quad \frac{T \vdash \text{enca}(u, \text{pub}(v)) \quad T \vdash \text{priv}(v)}{T \vdash u}$$

Intruder abilities

Composition rules

$$\frac{T \vdash u \quad T \vdash v}{T \vdash \langle u, v \rangle} \quad \frac{T \vdash u \quad T \vdash v}{T \vdash \text{enc}(u, v)} \quad \frac{T \vdash u \quad T \vdash v}{T \vdash \text{enca}(u, v)}$$



Decomposition rules

$$\frac{}{T \vdash u} u \in T \quad \frac{T \vdash \langle u, v \rangle}{T \vdash u} \quad \frac{T \vdash \langle u, v \rangle}{T \vdash v}$$

$$\frac{T \vdash \text{enc}(u, v) \quad T \vdash v}{T \vdash u} \quad \frac{T \vdash \text{enca}(u, \text{pub}(v)) \quad T \vdash \text{priv}(v)}{T \vdash u}$$

Deducibility relation

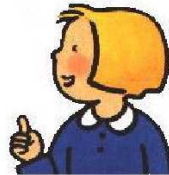
A term u is **deducible** from a set of terms T , denoted by $T \vdash u$, if there exists a proof tree witnessing this fact.

A simple protocol



$\langle \text{Bob}, k \rangle$

$\langle \text{Alice}, \text{enc}(s, k) \rangle$

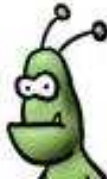
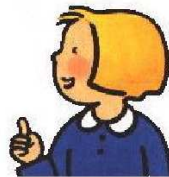


A simple protocol



$\langle \text{Bob}, k \rangle$

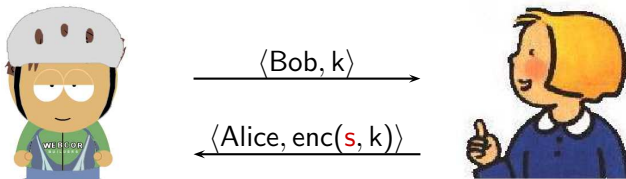
$\langle \text{Alice}, \text{enc}(s, k) \rangle$



Question ?

Can the attacker learn the secret s ?

A simple protocol



Answer : Of course, **Yes** !

$$\frac{\frac{\langle \text{Alice}, \text{enc}(s, k) \rangle}{\text{enc}(s, k)} \quad \frac{\langle \text{Bob}, k \rangle}{k}}{s}$$

Decision of the intruder problem

Given A set of messages S and a message m

Question Can the intruder learn m from S that is $S \vdash m$?

This problem is decidable in polynomial time. (left as exercise)

Decision of the intruder problem

Given A set of messages S and a message m

Question Can the intruder learn m from S that is $S \vdash m$?

This problem is decidable in polynomial time. (left as exercise)

Lemma (Locality)

If there is a proof of $S \vdash m$ then there is a proof that only uses the subterms of S and m .

Protocol description

Protocol :

$$A \rightarrow B : \{\text{pin}\}_{k_a}$$

$$B \rightarrow A : \{\{\text{pin}\}_{k_a}\}_{k_b}$$

$$A \rightarrow B : \{\text{pin}\}_{k_b}$$

A **protocol** is a **finite set of roles** :

- role $\Pi(1)$ corresponding to the 1st participant played by a talking to b :

$$\text{init} \xrightarrow{k_a} \text{enc}(\text{pin}, k_a)$$

$$\text{enc}(x, k_a) \rightarrow x.$$

Protocol description

Protocol :

$$A \rightarrow B : \{\text{pin}\}_{k_a}$$
$$B \rightarrow A : \{\{\text{pin}\}_{k_a}\}_{k_b}$$
$$A \rightarrow B : \{\text{pin}\}_{k_b}$$

A **protocol** is a **finite set of roles** :

- role $\Pi(1)$ corresponding to the 1st participant played by a talking to b :

$$\text{init} \xrightarrow{k_a} \text{enc}(\text{pin}, k_a)$$
$$\text{enc}(x, k_a) \rightarrow x.$$

- role $\Pi(2)$ corresponding to the 2nd participant played by b with a :

$$x \xrightarrow{k_b} \text{enc}(x, k_b)$$
$$\text{enc}(y, k_b) \rightarrow \text{stop}.$$

Secrecy via constraint solving [Millen et al]

Constraint systems are used to specify secrecy preservation under a particular, finite scenario.

Scenario

$$\begin{aligned} \text{rcv}(u_1) &\xrightarrow{N_1} \text{snd}(v_1) \\ \text{rcv}(u_2) &\xrightarrow{N_2} \text{snd}(v_2) \\ &\dots \\ \text{rcv}(u_n) &\xrightarrow{N_n} \text{snd}(v_n) \end{aligned}$$

Constraint System

$$\mathcal{C} = \begin{cases} T_0 \Vdash u_1 \\ T_0, v_1 \Vdash u_2 \\ \dots \\ T_0, v_1, \dots, v_n \Vdash s \end{cases}$$

where T_0 is the initial knowledge of the attacker.

Remark : Constraint Systems may be used more generally for trace-based properties, e.g. authentication.

Secrecy via constraint solving [Millen et al]

Constraint systems are used to specify secrecy preservation under a particular, finite scenario.

Scenario

$$\begin{aligned} \text{rcv}(u_1) &\xrightarrow{M_1} \text{snd}(v_1) \\ \text{rcv}(u_2) &\xrightarrow{M_2} \text{snd}(v_2) \\ &\dots \\ \text{rcv}(u_n) &\xrightarrow{M_n} \text{snd}(v_n) \end{aligned}$$

Constraint System

$$\mathcal{C} = \begin{cases} T_0 \Vdash u_1 \\ T_0, v_1 \Vdash u_2 \\ \dots \\ T_0, v_1, \dots, v_n \Vdash s \end{cases}$$

where T_0 is the initial knowledge of the attacker.

Solution of a constraint system

A substitution σ such that

for every $T \Vdash u \in \mathcal{C}$, $u\sigma$ is deducible from $T\sigma$, that is $u\sigma \vdash T\sigma$.

Example of a system constraint

$A \rightarrow B : \{\text{pin}\}_{k_a}$
 $B \rightarrow A : \{\{\text{pin}\}_{k_a}\}_{k_b}$ and the attacker initially knows $T_0 = \{\text{init}\}$.
 $A \rightarrow B : \{\text{pin}\}_{k_b}$

One possible associated constraint system is :

$$C = \left\{ \begin{array}{l} \{\text{init}\} \Vdash \text{init} \\ \{\text{init}, \{\text{pin}\}_{k_a}\} \Vdash \{x\}_{k_a} \\ \{\text{init}, \{\text{pin}\}_{k_a}, x\} \Vdash \text{pin} \end{array} \right.$$

Is there a solution ?

Example of a system constraint

$A \rightarrow B : \{\text{pin}\}_{k_a}$
 $B \rightarrow A : \{\{\text{pin}\}_{k_a}\}_{k_b}$ and the attacker initially knows $T_0 = \{\text{init}\}$.
 $A \rightarrow B : \{\text{pin}\}_{k_b}$

One possible associated constraint system is :

$$\mathcal{C} = \begin{cases} \{\text{init}\} \Vdash \text{init} \\ \{\text{init}, \{\text{pin}\}_{k_a}\} \Vdash \{x\}_{k_a} \\ \{\text{init}, \{\text{pin}\}_{k_a}, x\} \Vdash \text{pin} \end{cases}$$

Is there a solution ?

Of course yes, simply consider $x = \text{pin}$!

How to solve constraint system ?

$$\text{Given } \mathcal{C} = \begin{cases} T_0 \Vdash u_1 \\ T_0, v_1 \Vdash u_2 \\ \dots \\ T_0, v_1, \dots, v_n \Vdash u_{n+1} \end{cases}$$

Question Is there a solution σ of \mathcal{C} ?

An easy case : “solved constraint systems”

$$\text{Given } \mathcal{C} = \begin{cases} T_0 \Vdash x_1 \\ T_0, v_1 \Vdash x_2 \\ \dots \\ T_0, v_1, \dots, v_n \Vdash x_{n+1} \end{cases}$$

Question Is there a solution σ of \mathcal{C} ?

An easy case : “solved constraint systems”

$$\text{Given } \mathcal{C} = \begin{cases} T_0 \Vdash x_1 \\ T_0, v_1 \Vdash x_2 \\ \dots \\ T_0, v_1, \dots, v_n \Vdash x_{n+1} \end{cases}$$

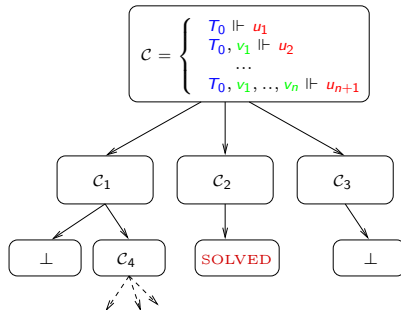
Question Is there a solution σ of \mathcal{C} ?

Of course yes!

Consider e.g. $\sigma(x_1) = \dots = \sigma(x_{n+1}) = t \in T_0$.

Decision procedure [Millen / Comon-Lundh]

Goal : Transformation of the constraints in order to obtain a solved constraint system.



C has a solution iff $C \rightsquigarrow C'$ with C' in solved form.

Intruder step

The intruder can built messages

$$R_5 : \mathcal{C} \wedge T \Vdash f(u, v) \rightsquigarrow \mathcal{C} \wedge T \Vdash u \wedge T \Vdash v$$

for $f \in \{\langle \rangle, \text{enc}, \text{enca}\}$

Intruder step

The intruder can build messages

$$R_5 : \mathcal{C} \wedge T \Vdash f(u, v) \rightsquigarrow \mathcal{C} \wedge T \Vdash u \wedge T \Vdash v \\ \text{for } f \in \{\langle \rangle, \text{enc}, \text{enca}\}$$

Example :

$$a, k \Vdash \text{enc}(\langle x, y \rangle, k) \rightsquigarrow \begin{array}{l} a, k \Vdash k \\ a, k \Vdash x \\ a, k \Vdash y \end{array}$$

Eliminating redundancies

$$k \Vdash x$$
$$k, \text{enc}(s, x) \Vdash s$$

The constraint $\text{enc}(s, x) \Vdash s$ will be satisfied as soon as $k \Vdash x$ is satisfied.

Eliminating redundancies

$$k \Vdash x$$

$$k, \text{enc}(s, x) \Vdash s$$

The constraint $\text{enc}(s, x) \Vdash s$ will be satisfied as soon as $k \Vdash x$ is satisfied.

$$R_1 : \mathcal{C} \wedge T \Vdash u \rightsquigarrow \mathcal{C} \quad \text{if } T \cup \{x \mid T' \Vdash x \in \mathcal{C}, T' \subsetneq T\} \Vdash u$$

Unsolvable constraints

$$R_4 : \mathcal{C} \wedge T \Vdash u \rightsquigarrow \perp \quad \text{if } \text{var}(T, u) = \emptyset \text{ and } T \not\vdash u$$

Example :

$$\begin{array}{l} \dots \\ a, \text{enc}(s, k) \Vdash s \quad \rightsquigarrow \quad \perp \\ \dots \end{array}$$

Guessing equalities

① Example : $k, \text{enc}(\text{enc}(x, k'), k) \Vdash \text{enc}(a, k')$

$$R_2 : \mathcal{C} \wedge T \Vdash u \rightsquigarrow_{\sigma} \mathcal{C}\sigma \wedge T\sigma \Vdash u\sigma \quad u' \in \text{st}(T) \\ \text{if } \sigma = \text{mgu}(u, u'), u, u' \notin \mathcal{X}, u \neq u'$$

Guessing equalities

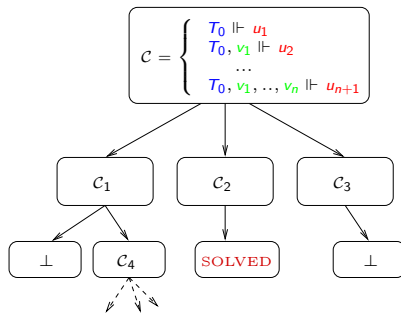
① Example : $k, \text{enc}(\text{enc}(x, k'), k) \Vdash \text{enc}(a, k')$

$$R_2 : \mathcal{C} \wedge T \Vdash u \rightsquigarrow_{\sigma} \mathcal{C}\sigma \wedge T\sigma \Vdash u\sigma \quad u' \in \text{st}(T) \\ \text{if } \sigma = \text{mgu}(u, u'), u, u' \notin \mathcal{X}, u \neq u'$$

② Example : $\text{enc}(s, \langle a, x \rangle), \text{enc}(\langle y, b \rangle, k), k \Vdash s$

$$R_3 : \mathcal{C} \wedge T \Vdash v \rightsquigarrow_{\sigma} \mathcal{C}\sigma \wedge T\sigma \Vdash v\sigma \quad u, u' \in \text{st}(T) \\ \text{if } \sigma = \text{mgu}(u, u'), u, u' \notin \mathcal{X}, u \neq u'$$

NP-procedure for solving constraint systems



Theorem

- C has a solution iff $C \rightsquigarrow C'$ with C' in solved form.
- \rightsquigarrow is terminating in polynomial time.

Example of tool : Avispa Platform

The screenshot displays the AVISPA web interface. The top section, titled "AVISPA - Automated Validation of Internet Security Protocols and Applications", shows the "Protocol" tab selected. The protocol is identified as "H.530 Symmetric security procedures for H.323 mobility in H.530". The purpose is to establish an authenticated (Diffie-Hellman) shared-key between a mobile terminal (MT) and a visited gate-keeper (VWK) who don't know each other, but who know an authentication facility (AuF) in MT's home domain. The reference is provided as "http://www.itsu.int/doc/recommendation.asp?typ=ALICE_B08".

The "Tools" section on the left lists several analysis tools: HUPSI, HUPSLW, and a group containing DRAC, RTSE, SATMC, and TRASP.

The main area shows an "msec ATTACK TRACE" for the role "H.530 hlp1". It details the interactions between three agents: Agent 1, Agent (a.3), and Agent (a.7). The trace shows a sequence of messages:

- Agent 1 sends $a.b.ni.CH1(1).exp(g.X(1)).fzz.a.a.aur.a.b.ni.CH1(1).exp(g.X(1))$ to Agent (a.3).
- Agent (a.3) sends $b.a.ni.x99.g.x92$ to Agent (a.7).
- Agent (a.7) sends $b.a.ni.x99.g.x92.a.g$ to Agent (a.3).
- Agent (a.3) sends $a.b.ni.CH1(1).exp(g.X(1)).fzz.a.a.aur.a.b.ni.CH1(1).exp(g.X(1))$ to Agent (a.7).
- Agent (a.7) sends $a.b.x99.CH2(3).exp(g.Y(2)).CH1(1).exp(g.X(1)).ni.fexp(g.Y(2)).a.b.x99.CH2(3).exp(g.Y(2)).CH1(1).exp(g.X(1)).ni$ to Agent (a.3).
- Agent (a.3) sends $b.a.CH2(3).x102.exp(g.Y(2)).b.a.CH2(3).x102$ to Agent (a.7).
- Agent (a.7) sends $a.b.x102.CH4(4).exp(g.Y(2)).a.b.x102.CH4(4)$ to Agent (a.3).

Collaborators

- LORIA, France
- DIST, Italy
- ETHZ, Switzerland
- Siemens, Germany

www.avispa-project.org

Limitations of this approach ?

Are you ready to use any protocol verified with this technique ?

Limitations of this approach ?

Are you ready to use any protocol verified with this technique ?

- Only a **finite scenario** is checked.
→ What happens if the protocol is used one more time ?
- The **underlying mathematical properties** of the primitives are abstracted away.

How to decide security for unlimited sessions ?

→ In general, it is **undecidable**!
(i.e. there exists **no** algorithm for checking e.g. secrecy)

How to prove undecidability ?

How to decide security for unlimited sessions ?

→ In general, it is **undecidable**!
(i.e. there exists **no** algorithm for checking e.g. secrecy)

How to prove undecidability ?

Post correspondence problem (PCP)

input $\{(u_i, v_i)\}_{1 \leq i \leq n}$, $u_i, v_i \in \Sigma^*$

output $\exists n, i_1, \dots, i_n \quad u_{i_1} \cdots u_{i_n} = v_{i_1} \cdots v_{i_n}$

Example : $\{(bab, b), (ab, aba), (a, baba)\}$

Solution ?

How to decide security for unlimited sessions ?

→ In general, it is **undecidable**!
(i.e. there exists **no** algorithm for checking e.g. secrecy)

How to prove undecidability ?

Post correspondence problem (PCP)

input $\{(u_i, v_i)\}_{1 \leq i \leq n}$, $u_i, v_i \in \Sigma^*$
output $\exists n, i_1, \dots, i_n \quad u_{i_1} \cdots u_{i_n} = v_{i_1} \cdots v_{i_n}$

Example : $\{(bab, b), (ab, aba), (a, baba)\}$

Solution ? → Yes, 1,2,3,1.

babababab
babababab

How to encode PCP in protocols?

Given $\{(u_i, v_i)\}_{1 \leq i \leq n}$, we construct the following protocol P :

$$\begin{array}{l}
 A \rightarrow B : \{\langle \overline{u_1}, \overline{v_1} \rangle\}_{K_{ab}}, \dots, \{\langle \overline{u_k}, \overline{v_k} \rangle\}_{K_{ab}} \\
 B : \{\langle x, y \rangle\}_{K_{ab}} \rightarrow A : \{\langle \overline{x}, \overline{u_1}, \overline{y}, \overline{v_1} \rangle\}_{K_{ab}}, \{s\} \{\langle \overline{x}, \overline{u_1}, \overline{x}, \overline{u_1} \rangle\}_{K_{ab}}, \\
 \dots, \{\langle \overline{x}, \overline{u_k}, \overline{y}, \overline{v_k} \rangle\}_{K_{ab}}, \{s\} \{\langle \overline{x}, \overline{u_k}, \overline{x}, \overline{u_k} \rangle\}_{K_{ab}}
 \end{array}$$

where $\overline{a_1 \cdot a_2 \cdots a_n}$ denotes the term $\langle \cdots \langle \langle a_1, a_2 \rangle, a_3 \rangle \cdots a_n \rangle$.

How to encode PCP in protocols ?

Given $\{(u_i, v_i)\}_{1 \leq i \leq n}$, we construct the following protocol P :

$$\begin{array}{l}
 A \rightarrow B : \{ \langle \overline{u_1}, \overline{v_1} \rangle \}_{K_{ab}}, \dots, \{ \langle \overline{u_k}, \overline{v_k} \rangle \}_{K_{ab}} \\
 B : \{ \langle x, y \rangle \}_{K_{ab}} \rightarrow A : \{ \langle \overline{x}, \overline{u_1}, \overline{y}, \overline{v_1} \rangle \}_{K_{ab}}, \{ s \} \{ \langle \overline{x}, \overline{u_1}, \overline{x}, \overline{u_1} \rangle \}_{K_{ab}}, \\
 \dots, \{ \langle \overline{x}, \overline{u_k}, \overline{y}, \overline{v_k} \rangle \}_{K_{ab}}, \{ s \} \{ \langle \overline{x}, \overline{u_k}, \overline{x}, \overline{u_k} \rangle \}_{K_{ab}}
 \end{array}$$

where $\overline{a_1 \cdot a_2 \cdots a_n}$ denotes the term $\langle \cdots \langle \langle a_1, a_2 \rangle, a_3 \rangle \cdots a_n \rangle$.

Then there is an attack on P iff there is a solution to the Post Correspondence Problem with entry $\{(u_i, v_i)\}_{1 \leq i \leq n}$.

How to circumvent undecidability ?

- Find **decidable subclasses** of protocols.
- Design **semi-decision procedure**, that works in practice
- ...

How to model an unbounded number of sessions ?

“For any x , if the agent A receives $\text{enc}(x, k_a)$ then A responds with x .”

→ Use of first-order logic.

Intruder

Horn clauses perfectly reflects the attacker **symbolic manipulations** on terms.



$$I(x), I(y) \Rightarrow I(\langle x, y \rangle) \quad \text{pairing}$$

$$I(x), I(y) \Rightarrow I(\{x\}_y) \quad \text{encryption}$$

$$I(\{x\}_y), I(y) \Rightarrow I(x) \quad \text{decryption}$$

$$I(\langle x, y \rangle) \Rightarrow I(x) \quad \text{projection}$$

$$I(\langle x, y \rangle) \Rightarrow I(y) \quad \text{projection}$$

Protocol

Protocol :

$$\begin{aligned} A \rightarrow B & : \{\text{pin}\}_{k_a} \\ B \rightarrow A & : \{\{\text{pin}\}_{k_a}\}_{k_b} \\ A \rightarrow B & : \{\text{pin}\}_{k_b} \end{aligned}$$

Horn clauses :

$$\begin{aligned} & \Rightarrow I(\{\text{pin}\}_{k_a}) \\ I(x) & \Rightarrow I(\{x\}_{k_b}) \\ I(\{x\}_{k_a}) & \Rightarrow I(x) \end{aligned}$$

Protocol

Protocol :

$$\begin{aligned} A \rightarrow B & : \{\text{pin}\}_{k_a} \\ B \rightarrow A & : \{\{\text{pin}\}_{k_a}\}_{k_b} \\ A \rightarrow B & : \{\text{pin}\}_{k_b} \end{aligned}$$

Horn clauses :

$$\begin{aligned} & \Rightarrow I(\{\text{pin}\}_{k_a}) \\ I(x) & \Rightarrow I(\{x\}_{k_b}) \\ I(\{x\}_{k_a}) & \Rightarrow I(x) \end{aligned}$$

Secrecy property is a **reachability** (accessibility) property

$$\neg I(\text{pin})$$

Then there exists an attack iff the set of formula corresponding to
 Intruder manipulations + protocol + property
 is **NOT** satisfiable.

How to decide satisfiability ?

→ Resolution techniques

Some vocabulary

First order logic

Atoms $P(t_1, \dots, t_n)$ where t_i are terms, P is a predicate

Literals $P(t_1, \dots, t_n)$ or $\neg P(t_1, \dots, t_n)$

closed under $\vee, \wedge, \neg, \exists, \forall$

Clauses : Only universal quantifiers

Horn Clauses : at most one positive literal

$$A_1, \dots, A_n \Rightarrow B$$

where A_i, B are atoms.

Binary resolution

A, B are atoms and C, D are clauses.

An intuitive rule

$$\frac{A \Rightarrow C \quad A}{C}$$

In other words

$$\frac{\neg A \vee C \quad A}{C}$$

Binary resolution

A, B are atoms and C, D are clauses.

An intuitive rule

$$\frac{A \Rightarrow C \quad A}{C}$$

In other words

$$\frac{\neg A \vee C \quad A}{C}$$

Generalizing

$$\frac{\neg A \vee C \quad B}{C\theta} \quad \theta = mgu(A, B) \quad (\text{i.e. } A\theta = B\theta)$$

Binary resolution

A, B are atoms and C, D are clauses.

An intuitive rule

$$\frac{A \Rightarrow C \quad A}{C}$$

In other words

$$\frac{\neg A \vee C \quad A}{C}$$

Generalizing

$$\frac{\neg A \vee C \quad B}{C\theta} \quad \theta = mgu(A, B) \quad (\text{i.e. } A\theta = B\theta)$$

Generalizing a bit more

$$\frac{\neg A \vee C \quad B \vee D}{C\theta \vee D\theta} \quad \theta = mgu(A, B) \quad \text{Binary resolution}$$

Binary resolution and Factorization

$$\frac{\neg A \vee C \quad B \vee D}{C\theta \vee D\theta} \theta = \text{mgu}(A, B) \quad \text{Binary resolution}$$

$$\frac{A \vee B \vee C}{A\theta \vee C\theta} \theta = \text{mgu}(A, B) \quad \text{Factorisation}$$

Theorem (Soundness and Completeness)

Binary resolution and factorisation are *sound and refutationally complete*,

i.e. a set of clauses C is not satisfiable if and only if \perp (the empty clause) can be obtained from C by binary resolution and factorisation.

Exercise : Why do we need the factorisation rule?

Example

$$\mathcal{C} = \{ \neg I(s), \quad I(k_1), \quad I(\{s\}_{\langle k_1, k_1 \rangle}), \\ I(\{x\}_y), I(y) \Rightarrow I(x), \quad I(x), I(y) \Rightarrow I(\langle x, y \rangle) \}$$

$$\frac{\frac{\frac{I(\{s\}_{\langle k_1, k_1 \rangle}) \quad I(\{x\}_y), I(y) \Rightarrow I(x)}{I(\langle k_1, k_1 \rangle) \Rightarrow s} \quad \frac{\frac{I(k_1) \quad I(x), I(y) \Rightarrow I(\langle x, y \rangle)}{I(y) \Rightarrow I(\langle k_1, y \rangle)}{I(\langle k_1, k_1 \rangle)}}{\frac{\neg I(s) \quad I(s)}{\perp}}}$$

But it is not terminating !

$$\begin{array}{c}
 \frac{I(s) \quad I(x), I(y) \Rightarrow I(\langle x, y \rangle)}{I(s) \quad I(y) \Rightarrow I(\langle s, y \rangle)} \\
 \frac{I(y) \Rightarrow I(\langle s, y \rangle) \quad I(\langle s, s \rangle)}{I(y) \Rightarrow I(\langle s, y \rangle) \quad I(\langle s, \langle s, s \rangle \rangle)} \\
 \frac{I(y) \Rightarrow I(\langle s, y \rangle) \quad I(\langle s, \langle s, s \rangle \rangle)}{I(\langle s, \langle s, \langle s, s \rangle \rangle \rangle)} \\
 \dots
 \end{array}$$

→ This does not yield any decidability result.

Ordered Binary resolution and Factorization

Let $<$ be any order on clauses.

$$\frac{\neg A \vee C \quad B \vee D \quad \theta = \text{mgu}(A, B)}{C\theta \vee D\theta} \quad A\theta \not< C\theta \vee D\theta \quad \text{Ordered binary resolution}$$

$$\frac{A \vee B \vee C \quad \theta = \text{mgu}(A, B)}{A\theta \vee C\theta} \quad A\theta \not< C\theta \quad \text{Ordered factorisation}$$

Ordered Binary resolution and Factorization

Let $<$ be any order on clauses.

$$\frac{\neg A \vee C \quad B \vee D}{C\theta \vee D\theta} \quad \theta = \text{mgu}(A, B) \quad \text{Ordered binary resolution}$$

$A\theta \not< C\theta \vee D\theta$

$$\frac{A \vee B \vee C}{A\theta \vee C\theta} \quad \theta = \text{mgu}(A, B) \quad \text{Ordered factorisation}$$

$A\theta \not< C\theta$

Theorem (Soundness and Completeness)

Ordered binary resolution and factorisation are sound and refutationally complete provided that $<$ is liftable

$$\forall A, B, \theta \quad A < B \Rightarrow A\theta < B\theta$$

Examples of liftable orders

$$\forall A, B, \theta \quad A < B \Rightarrow A\theta < B\theta$$

First example : subterm order

$P(t_1, \dots, t_n) < Q(u_1, \dots, u_k)$ iff any t_i is a subterm of u_1, \dots, u_k

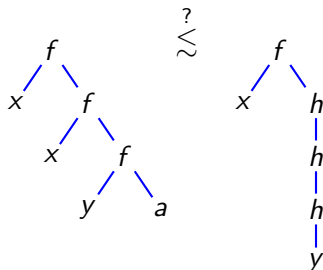
→ extended to clauses as follows : $C_1 < C_2$ iff any literal of C_1 is smaller than some literal of C_2 .

Exercise : Show that \mathcal{C} is not satisfiable by **ordered resolution (and factorisation)**.

Examples of liftable orders - continued

Second example : $P(t_1, \dots, t_n) \lesssim Q(u_1, \dots, u_k)$ iff

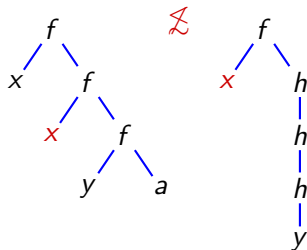
- ① $\text{depth}(P(t_1, \dots, t_n)) \leq \text{depth}(Q(u_1, \dots, u_k))$
- ② For any variable x ,
 $\text{depth}_x(P(t_1, \dots, t_n)) \leq \text{depth}_x(Q(u_1, \dots, u_k))$



Examples of liftable orders - continued

Second example : $P(t_1, \dots, t_n) \lesssim Q(u_1, \dots, u_k)$ iff

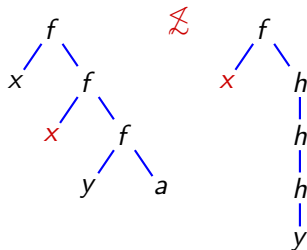
- ① $\text{depth}(P(t_1, \dots, t_n)) \leq \text{depth}(Q(u_1, \dots, u_k))$
- ② For any variable x ,
 $\text{depth}_x(P(t_1, \dots, t_n)) \leq \text{depth}_x(Q(u_1, \dots, u_k))$



Examples of liftable orders - continued

Second example : $P(t_1, \dots, t_n) \lesssim Q(u_1, \dots, u_k)$ iff

- 1 $\text{depth}(P(t_1, \dots, t_n)) \leq \text{depth}(Q(u_1, \dots, u_k))$
- 2 For any variable x ,
 $\text{depth}_x(P(t_1, \dots, t_n)) \leq \text{depth}_x(Q(u_1, \dots, u_k))$



Exercise : Show that $\forall A, B, \theta \quad A \lesssim B \Rightarrow A\theta \lesssim B\theta$

Back to protocols

Intruder clauses are of the form

$$\pm I(f(x_1, \dots, x_n)), \pm I(x_i), \pm I(x_j)$$

Protocol clauses

$$\Rightarrow I(\{\text{pin}\}_{k_a})$$

$$I(x) \Rightarrow I(\{x\}_{k_b})$$

$$I(\{x\}_{k_a}) \Rightarrow I(x)$$

At most one variable per clause!

Back to protocols

Intruder clauses are of the form

$$\pm I(f(x_1, \dots, x_n)), \pm I(x_i), \pm I(x_j)$$

Protocol clauses

$$\Rightarrow I(\{\text{pin}\}_{k_a})$$

$$I(x) \Rightarrow I(\{x\}_{k_b})$$

$$I(\{x\}_{k_a}) \Rightarrow I(x)$$

At most one variable per clause!

Theorem

Given a set \mathcal{C} of clauses such that each clause of \mathcal{C}

- either contains at most one variable
- or is of the form $\pm I(f(x_1, \dots, x_n)), \pm I(x_i), \pm I(x_j)$

Then ordered (\lesssim) binary resolution and factorisation is terminating.

Decidability for an unbounded number of sessions

Corollary

For any protocol that can be encoded with clauses of the previous form, then checking secrecy is decidable.

But how to deal with protocols that need more than one variable per clause?

ProVerif

Developed by Bruno Blanchet, Paris, France.

- No restriction on the clauses
- Implements a **sound semi-decision procedure** (that may not terminate).
- Based on a resolution strategy **well adapted to protocols**.
- **performs very well in practice!**
 - Works on **most of existing protocols** in the literature
 - Is also used on **industrial protocols** (e.g. certified email protocol, JFK, Plutus filesystem)

What formal methods allow to do ?

- In general, secrecy preservation is **undecidable**.

What formal methods allow to do ?

- In general, secrecy preservation is **undecidable**.
- For a **bounded number of sessions**, secrecy is **co-NP-complete**
[RusinowitchTurvani CSFW01]
→ **several tools for detecting attacks** (Casper, Avispa platform...)

What formal methods allow to do ?

- In general, secrecy preservation is **undecidable**.
- For a **bounded number of sessions**, secrecy is **co-NP-complete** [RusinowitchTurvani CSFW01]
→ **several tools for detecting attacks** (Casper, Avispa platform...)
- For an unbounded number of sessions
 - for **one-copy protocols**, secrecy is **DEXPTIME-complete** [CortierComon RTA03] [SeildVerma LPAR04]
 - for **message-length bounded protocols**, secrecy is **DEXPTIME-complete** [Durgin et al FMSP99] [Chevalier et al CSL03]→ **some tools for proving security** (ProVerif, EVA Platform)

Limitations of this approach ?

Are you ready to use any protocol verified with this technique ?

- Only a **finite scenario** is checked.
→ What happens if the protocol is used one more time ?
- **The underlying mathematical properties of the primitives are abstracted away.**

Motivation

Back to our running example :

$$A \rightarrow B : \{\text{pin}\}_{k_a}$$

$$B \rightarrow A : \{\{\text{pin}\}_{k_a}\}_{k_b}$$

$$A \rightarrow B : \{\text{pin}\}_{k_b}$$

We need the equation for the commutativity of encryption

$$\{\{z\}_x\}_y = \{\{z\}_y\}_x$$

Some other examples

Encryption-Decryption theory

$$\text{dec}(\text{enc}(x, y), y) = x \quad \pi_1(\langle x, y \rangle) = x \quad \pi_2(\langle x, y \rangle) = y$$

EXclusive Or

$$\begin{aligned} x \oplus (y \oplus z) &= z & x \oplus y &= y \oplus x \\ x \oplus x &= 0 & x \oplus 0 &= x \end{aligned}$$

Diffie-Hellmann

$$\text{exp}(\text{exp}(z, x), y) = \text{exp}(\text{exp}(z, y), x)$$

E-voting protocols

First phase :

$V \rightarrow A : \text{sign}(\textit{blind}(\textit{vote}, r), V)$

$A \rightarrow V : \text{sign}(\textit{blind}(\textit{vote}, r), A)$

Voting phase :

$V \rightarrow C : \text{sign}(\textit{vote}, A)$

...



Equational theory for blind signatures

[Kremer Ryan 05]

$$\begin{aligned}\text{checksign}(\text{sign}(x, y), \text{pk}(y)) &= x \\ \text{unblind}(\text{blind}(x, y), y) &= x \\ \text{unblind}(\text{sign}(\text{blind}(x, y), z), y) &= \text{sign}(x, z)\end{aligned}$$

Deduction

$$\frac{}{T \vdash_E M} M \in T$$

$$\frac{T \vdash_E M_1 \quad \dots \quad T \vdash_E M_k}{T \vdash_E f(M_1, \dots, M_k)} f \in \Sigma$$

$$\frac{T \vdash M}{T \vdash M'} M =_E M'$$

Deduction

$$\frac{}{T \vdash_E M} M \in T \qquad \frac{T \vdash_E M_1 \quad \dots \quad T \vdash_E M_k}{T \vdash_E f(M_1, \dots, M_k)} f \in \Sigma$$

$$\frac{T \vdash M}{T \vdash M'} M =_E M'$$

Example : $E := \text{dec}(\text{enc}(x, y), y) = x$ and $T = \{\text{enc}(\text{secret}, k), k\}$.

$$\frac{\frac{\frac{}{T \vdash \text{enc}(\text{secret}, k)}}{T \vdash \text{dec}(\text{enc}(\text{secret}, k), k)}}{T \vdash \text{secret}} \quad \frac{}{T \vdash k} \quad f \in \Sigma \quad \text{dec}(\text{enc}(x, y), y) = x$$

Rewriting systems

For analyzing equational theories, we (try to) associate to E a finite convergent rewriting system \mathcal{R} such that :

$$u =_E v \quad \text{iff} \quad u \downarrow = v \downarrow$$

Definition (Characterization of the deduction relation)

Let t_1, \dots, t_n and u be terms in normal form.

$$\{t_1, \dots, t_n\} \vdash u \quad \text{iff} \quad \exists C \text{ s.t. } C[t_1, \dots, t_n] \rightarrow^* u$$

(Also called Cap Intruder problem [Narendran et al])

Some results with equational theories

	Security problem	
	Bounded number of sessions	Unbounded number of sessions
Commutative encryption	<i>co-NP-complete</i> [CKRT04]	Ping-pong protocols : <i>co-NP-complete</i> [Turvani04]
Exclusive Or	<i>Decidable</i> [CS03,CKRT03]	One copy - No nonces : <i>Decidable</i> [CLC03] Two-way automata - No nonces : <i>Decidable</i> [Verma03]
Abelian Groups	<i>Decidable</i> [Shmatikov04]	Two-way automata - No nonces : <i>Decidable</i> [Verma03]
Prefix encryption	<i>co-NP-complete</i> [CKRT03]	
Abelian Groups and Modular Exponentiation	General case : <i>Decidable</i> [Shmatikov04] Restricted protocols : <i>co-NP-complete</i> [CKRT03]	AC properties of the Modular Exponentiation No nonces : <i>Semi-Decision Procedure</i> [GLRV04]

And now are you ready to use any protocol verified with these techniques ?

Assuming :

- Analysis for an unbounded number of sessions
- With equational theories

Outline of the talk

- 1 Introduction on security protocols
 - Context
 - Security Protocols : how does it work ?
 - Commutative encryption (RSA)
 - Needham-Schroeder Example
- 2 Formal models
 - Messages
 - Intruder
 - Protocol
 - Solving constraint systems
- 3 Going further
 - Undecidability
 - Horn clauses
 - Adding equational theories
 - Some results
- 4 Towards more guarantees
 - Cryptographic models
 - Linking Formal and cryptographic models
 - Extension to indistinguishability
 - Conclusion

Specificity of cryptographic models

- Messages are bitstrings
- Real encryption algorithm
- Real signature algorithm
- General and powerful adversary

→ **very little abstract model**

Encryption : the old time

- Caesar encryption : $A \rightarrow E, B \rightarrow F, C \rightarrow G, \dots$
- Cypher Disk (Léone Battista Alberti 1466)



Encryption : the old time

- Caesar encryption : $A \rightarrow E, B \rightarrow F, C \rightarrow G, \dots$
- Cypher Disk (Léone Battista Alberti 1466)



→ subject to statistical analysis (Analyzing letter frequencies)

Encryption : mechanized time

Automatic substitutions and permutations



Enigma

Encryption nowadays

→ Based on algorithmically hard problems.

RSA Function $n = pq$, p et q primes.

e : public exponent

- $x \mapsto x^e \pmod n$ easy (cubic)
- $y = x^e \mapsto x \pmod n$ difficult
 $x = y^d$ où $d = e^{-1} \pmod{\phi(n)}$

Encryption nowadays

→ Based on algorithmically hard problems.

RSA Function $n = pq$, p et q primes.

e : public exponent

- $x \mapsto x^e \pmod n$ easy (cubic)
- $y = x^e \mapsto x \pmod n$ difficult
 $x = y^d$ où $d = e^{-1} \pmod{\phi(n)}$

Diffie-Hellman Problem

- Given $A = g^a$ and $B = g^b$,
- Compute $\text{DH}(A, B) = g^{ab}$

Encryption nowadays

→ Based on algorithmically hard problems.

RSA Function $n = pq$, p et q primes.

e : public exponent

- $x \mapsto x^e \pmod n$ easy (cubic)
- $y = x^e \mapsto x \pmod n$ difficult
 $x = y^d$ où $d = e^{-1} \pmod{\phi(n)}$

Diffie-Hellman Problem

- Given $A = g^a$ and $B = g^b$,
- Compute $\text{DH}(A, B) = g^{ab}$

→ Based on hardness of integer factorization.

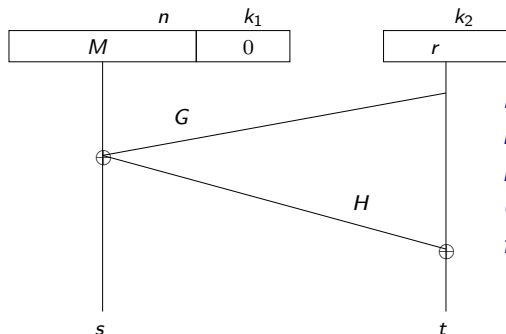
Estimations for integer factorization

Module (bits)	Operations (in \log_2)
512	58
1024	80
2048	111
4096	149
8192	156

$\approx 2^{60}$ years

→ Lower bound for RSA and Diffie-Hellman.

How does an encryption algorithm look like? Example : OAEP [Bellare Rogaway]



M : plaintext of length n
 r : randomness of length k_0
 G, H : hash function
 f_k : trapdoor function

$$E_K(x; r) = f_K(s||t)$$

Cryptographic models

Encryption is only one component of cryptographic models

- Cryptographic primitives : encryption, signatures, ...
- Protocol model
- Adversary
- Security notions

Setting for cryptographic protocols

Protocol :

- Message exchange program
- using cryptographic primitives

Adversary \mathcal{A} : any **probabilistic polynomial Turing machine**, *i.e.* any probabilistic polynomial program.

- **polynomial** : captures what is feasible
- **probabilistic** : the adversary may try to guess some information



Definition of secrecy preservation

→ Several notions of secrecy :

One-Wayness : The probability for an adversary \mathcal{A} to compute the secret s against a protocol \mathcal{P} is **negligible** (smaller than any inverse of polynomial).

$$\forall p \text{ polynomial } \exists \eta_0 \forall \eta \geq \eta_0 \Pr_{m,r}^{\eta}[\mathcal{A}(\mathcal{P}_K) = s] \leq \frac{1}{p(\eta)}$$

η : security parameter = key length

Not strong enough !

- The adversary may be able to compute half of the secret message.
- There is no guarantee in case that some partial information on the secret is known.



Not strong enough !

- The adversary may be able to compute half of the secret message.
- There is no guarantee in case that some partial information on the secret is known.



→ Introduction of a notion of indistinguishability.

Indistinguishability

The **secrecy** of s is defined through the following game :

- Two values n_0 and n_1 are randomly generated instead of s ;
- The adversary interacts with the protocol where s is replaced by n_b , $b \in \{0, 1\}$;
- We give the pair (n_0, n_1) to the adversary ;
- The adversary gives b' ,

The data s is secret if $\Pr[b = b'] - \frac{1}{2}$ is a **negligible** function.

A typical cryptographic proof

- 1 Assume that some algorithmic problem P is difficult (E.g. RSA or integer factorization or Discrete Log or CDH, DDH, ...)
- 2 Suppose that a (polynomial probabilistic) adversary \mathcal{A} breaks the protocol security with non negligible probability

A typical cryptographic proof

- 1 Assume that some algorithmic problem P is difficult (E.g. RSA or integer factorization or Discrete Log or CDH, DDH, ...)
- 2 Suppose that a (polynomial probabilistic) adversary \mathcal{A} breaks the protocol security with non negligible probability
- 3 Build out of \mathcal{A} an adversary \mathcal{B} that solves P .

A typical cryptographic proof

- 1 Assume that some algorithmic problem P is difficult (E.g. RSA or integer factorization or Discrete Log or CDH, DDH, ...)
- 2 Suppose that a (polynomial probabilistic) adversary \mathcal{A} breaks the protocol security with non negligible probability
- 3 Build out of \mathcal{A} an adversary \mathcal{B} that solves P .
- 4 Conclude that the protocol is secure provided P is difficult.

Formal and Cryptographic approaches

	Formal approach	Cryptographic approach
Messages	terms	bitstrings
Encryption	idealized	algorithm
Adversary	idealized	any polynomial algorithm
Secrecy property	reachability-based property	indistinguishability
Guarantees	unclear	strong
Protocol	may be complex	usually simpler

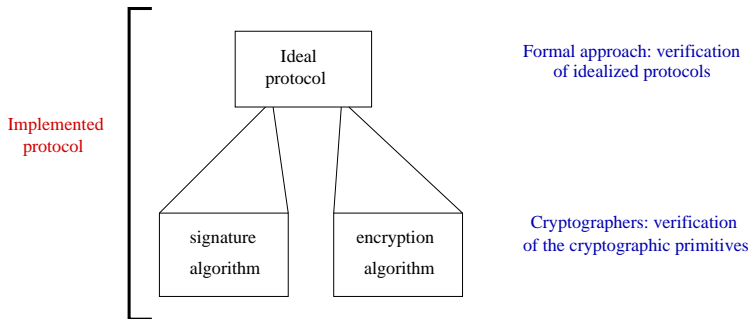
Formal and Cryptographic approaches

	Formal approach	Cryptographic approach
Messages	terms	bitstrings
Encryption	idealized	algorithm
Adversary	idealized	any polynomial algorithm
Secrecy property	reachability-based property	indistinguishability
Guarantees	unclear	strong
Protocol	may be complex	usually simpler
Proof	automatic	by hand, tedious and error-prone

Link between the two approaches ?

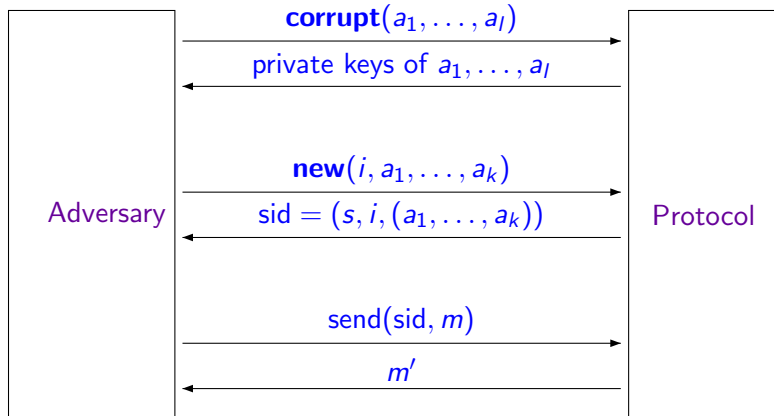
Composition of the two approaches

Automatic cryptographically sound proofs



A common setting

Same setting in formal and cryptographic models



Formal Intruder Deduction Rules

$$\frac{S \vdash m_1 \quad S \vdash m_2}{S \vdash \langle m_1, m_2 \rangle}$$

$$\frac{S \vdash \langle m_1, m_2 \rangle}{S \vdash m_i} \quad i \in \{1, 2\}$$

$$\frac{S \vdash \text{ek}(b) \quad S \vdash m \quad i \in \mathbb{N}}{S \vdash \{m\}_{\text{ek}(b)}^{\text{adv}(i)}}$$

$$\frac{S \vdash \{m\}_{\text{ek}(b)}^i \quad S \vdash \text{dk}(b)}{S \vdash m}$$

$$\frac{S \vdash \text{sk}(b) \quad S \vdash m \quad i \in \mathbb{N}}{S \vdash [m]_{\text{sk}(b)}^{\text{adv}(i)}}$$

$$\frac{S \vdash [m]_{\text{sk}(b)}^i}{S \vdash m}$$

Result : Soundness of trace properties

Theorem (extension of [Micciancio Warinschi TCC'04])

Every concrete trace is the image of a valid formal trace, except with negligible probability.

Result : Soundness of trace properties

Theorem (extension of [Micciancio Warinschi TCC'04])

Every concrete trace is the image of a valid formal trace, except with negligible probability.

Corollary :

Let Π be protocol, P^s an arbitrary predicate on formal traces and P^c its corresponding predicate on concrete traces.

Then $\Pi \models^s P^s$ implies $\Pi \models^c P^c$.

Result : Soundness of trace properties

Theorem (extension of [Micciancio Warinschi TCC'04])

Every concrete trace is the image of a valid formal trace, except with negligible probability.

Corollary :

Let Π be protocol, P^s an arbitrary predicate on formal traces and P^c its corresponding predicate on concrete traces.

Then $\Pi \models^s P^s$ implies $\Pi \models^c P^c$.

Applications : authentication, secrecy, ...

Hypotheses on the Implementation

- **encryption** : IND-CCA2
→ the adversary cannot distinguish between $\{n_0\}_k$ and $\{n_1\}_k$ even if he has access to encryption and decryption oracles.
- **signature** : randomized and existentially unforgeable under chosen-message attack *i.e.* one can not produce a valid pair (m, σ)
- **parsing** :
 - each bit-string has a label which indicates his type (identity, nonce, key, signature, ...)
 - one can retrieve the (public) encryption key from an encrypted message.
 - one can retrieve the signed message from the signature

▶ skip the proof

Proof idea

Proof technique : Reducing the protocol security to the robustness of the primitives (which itself reduces to hardness of algorithmic problem like integer factorization).

\mathcal{A} breaks $\mathcal{P} \Rightarrow \mathcal{A}'$ breaks $\{ \}$ or sign

Proof idea

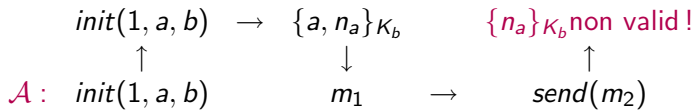
Proof technique : Reducing the protocol security to the robustness of the primitives (which itself reduces to hardness of algorithmic problem like integer factorization).

\mathcal{A} breaks $\mathcal{P} \Rightarrow \mathcal{A}'$ breaks $\{ \}$ or sign

Example : If a computational (concrete) adversary \mathcal{A} is able to compute $\{n_a\}_{K_a}$ out of $\{ \langle A, n_a \rangle \}_{K_a}$,
Then we can build an adversary \mathcal{A}' that breaks the encryption $\{ \}_{K_a}$.

Proof idea

Key result : every concrete trace is the image of a valid formal trace, except with negligible probability.



Proof idea

Key result : every concrete trace is the image of a valid formal trace, except with negligible probability.

$$\begin{array}{ccccc}
 & \text{init}(1, a, b) & \rightarrow & \{a, n_a\}_{K_b} & & \{n_a\}_{K_b} \text{ non valid !} \\
 & \uparrow & & \downarrow & & \uparrow \\
 \mathcal{A}: & \text{init}(1, a, b) & & m_1 & \rightarrow & \text{send}(m_2)
 \end{array}$$

Using the adversary \mathcal{A} , we build an adversary \mathcal{A}' that breaks encryption.

$$\mathcal{A}' : (\langle a, n_a^0 \rangle, \langle a, n_a^1 \rangle) \rightarrow \text{encryption oracle} \rightarrow \{a, n_a^\alpha\}_{K_b}$$

Proof idea

Key result : every concrete trace is the image of a valid formal trace, except with negligible probability.

$$\begin{array}{ccccc}
 & \text{init}(1, a, b) & \rightarrow & \{a, n_a\}_{K_b} & & \{n_a\}_{K_b} \text{ non valid !} \\
 & \uparrow & & \downarrow & & \uparrow \\
 \mathcal{A}: & \text{init}(1, a, b) & & m_1 & \rightarrow & \text{send}(m_2)
 \end{array}$$

Using the adversary \mathcal{A} , we build an adversary \mathcal{A}' that breaks encryption.

$$\begin{aligned}
 \mathcal{A}' : \quad & (\langle a, n_a^0 \rangle, \langle a, n_a^1 \rangle) \rightarrow \text{encryption oracle} \rightarrow \{a, n_a^\alpha\}_{K_b} \\
 & \rightarrow \mathcal{A} \rightarrow \{n_a^\alpha\}_{K_b}
 \end{aligned}$$

Proof idea

Key result : every concrete trace is the image of a valid formal trace, except with negligible probability.

$$\begin{array}{ccccc}
 & \text{init}(1, a, b) & \rightarrow & \{a, n_a\}_{K_b} & & \{n_a\}_{K_b} \text{ non valid !} \\
 & \uparrow & & \downarrow & & \uparrow \\
 \mathcal{A}: & \text{init}(1, a, b) & & m_1 & \rightarrow & \text{send}(m_2)
 \end{array}$$

Using the adversary \mathcal{A} , we build an adversary \mathcal{A}' that breaks encryption.

$$\begin{array}{l}
 \mathcal{A}' : \quad (\langle a, n_a^0 \rangle, \langle a, n_a^1 \rangle) \rightarrow \text{encryption oracle} \rightarrow \{a, n_a^\alpha\}_{K_b} \\
 \rightarrow \mathcal{A} \rightarrow \{n_a^\alpha\}_{K_b} \rightarrow \text{decryption oracle} \rightarrow n_a^\alpha \rightarrow \alpha
 \end{array}$$

Correspondence of secrecy properties

Theorem

Symbolic secrecy implies cryptographic indistinguishability.

- For protocols with only **public key encryption, signatures and nonces**
- Provided the public key encryption and the signature algorithms verify **strong existing cryptographic properties** (IND-CCA2, existentially unforgeable),



Conclusion

Formal methods form a powerful approach for analyzing security protocols

- Makes use of classical techniques in formal methods : term algebra, equational theories, clauses and resolution techniques, tree automata, etc.
⇒ Many decision procedures
- Several automatic tools
 - For successfully detecting attacks on protocols (e.g. Casper, Avispa)
 - For proving security for an arbitrary number of sessions (e.g. ProVerif)
- Provides cryptographic guarantees under classical assumptions on the implementation of the primitives

Some current directions of research

• Enriching the symbolic model

- Considering more equational theories (e.g. theories for e-voting protocols)
- Adding more complex structures for data (list, XML, ...)
- Considering recursive protocols (e.g. group protocol) where the number of message exchanges in a session is not fixed
- Proving more complex security properties like equivalence-based properties (e.g. for anonymity or e-voting protocols)

• With cryptographic guarantees

- Combining formal and cryptographic models for more complex primitives and security properties.
- How far can we go ?
- Is it possible to consider weaker cryptographic primitives ?