# Can I get my security proof for free?

Véronique Cortier, CNRS

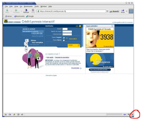
July 20th, 2010

SRM seminar

# Context : cryptographic protocols

Cryptographic protocols are widely used in everyday life.

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



# Security properties

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 ! !)
- and many other properties...

**Goal**

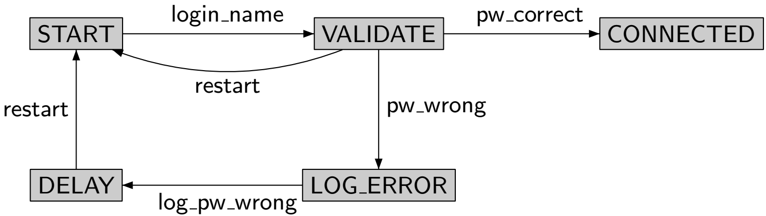
Check whether a protocol achieves its desired properties.

# Outline of the talk

1. Formal analysis of security protocols
   - Context
   - Modeling protocols
   - Solving constraint systems
   - Horn clauses
2. Composing protocols
   - Parallel composition
   - General composition
3. Towards more guarantees
   - Cryptographic models
   - Linking Formal and cryptographic models
   - Limitations
4. Conclusion

# Modeling protocol : a first approach

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



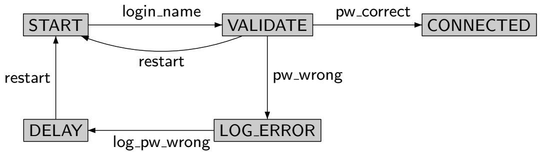
# Difficulty

Presence of an attacker

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

$\Rightarrow$ The system is infinitely branching

# 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.

$\rightarrow$ It is important to have a tight modeling of the messages.

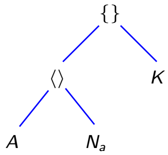
# Messages

Messages are abstracted by terms.

| | | |
|---|---|---|
| Agents : $a, b, \ldots$ | Nonces : $n_1, n_2, \ldots$ | Keys : $k_1, k_2, \ldots$ |
| Cyphertext : $\mathrm{enc}(m, k)$ | Concatenation : $\mathrm{pair}(m_1, m_2)$ | |

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

$$\mathrm{enc}(\mathrm{pair}(A, N_a), K)$$




Intuition : only the structure of the message is kept.

# Intruder abilities

## Composition rules

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

# Intruder abilities

## Composition rules

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

## Decomposition rules

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

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

# Intruder abilities

## Composition rules

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

## Decomposition rules

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

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

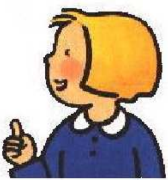
### Deducibility relation

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

# A simple protocol

$$\xrightarrow{\langle \mathsf{Bob}, \mathsf{k} \rangle}$$

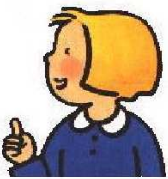
$$\xleftarrow{\langle \mathsf{Alice}, \mathsf{enc}(\color{red}{\mathsf{s}}, \mathsf{k}) \rangle}$$

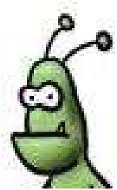


# A simple protocol


$$\xrightarrow{\langle \mathrm{Bob}, \mathrm{k} \rangle}$$

$$\xleftarrow{\langle \mathrm{Alice}, \mathrm{enc}(\textcolor{red}{\mathrm{s}}, \mathrm{k}) \rangle}$$





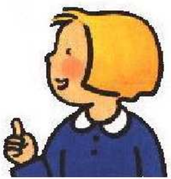
**Question ?**

Can the attacker learn the secret s ?

# A simple protocol

$$\xrightarrow{\langle \mathsf{Bob}, \mathsf{k}\rangle}$$

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



Answer : Of course, Yes !

$$\dfrac{\dfrac{\langle \mathsf{Alice}, \mathsf{enc}(\mathsf{s}, \mathsf{k})\rangle}{\mathsf{enc}(\mathsf{s}, \mathsf{k})} \qquad \dfrac{\langle \mathsf{Bob}, \mathsf{k}\rangle}{\mathsf{k}}}{\mathsf{s}}$$

# 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.

**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 :

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

A protocol is a finite set of roles :

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

$$\begin{array}{rcl} \text{init} & \xrightarrow{k_a} & \text{enc}(\text{pin}, k_a) \\ \text{enc}(x, k_a) & \rightarrow & x. \end{array}$$

# Protocol description

Protocol :

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

A protocol is a finite set of roles :

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

$$\begin{array}{rcl} \text{init} & \xrightarrow{k_a} & \text{enc}(\text{pin}, k_a) \\ \text{enc}(x, k_a) & \rightarrow & x. \end{array}$$

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

$$\begin{array}{rcl} x & \xrightarrow{k_b} & \text{enc}(x, k_b) \\ \text{enc}(y, k_b) & \rightarrow & \text{stop}. \end{array}$$

# Secrecy via constraint solving [Millen et al]

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

**Scenario**

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

**Constraint System**

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

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**

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

**Constraint System**

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

where $T_0$ is the initial knowledge of the attacker.

**Solution of a constraint system**

A substitution $\sigma$ such that

*for any* $T \Vdash u_i \in \mathcal{C}$, $u_i\sigma$ *is deducible from* $T\sigma$, *i.e.* $u_i\sigma \vdash T\sigma$.

# Example of a system constraint

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

and the attacker initially knows $T_0 = \{\text{init}\}$.

One possible associated constraint system is :

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

Is there a solution ?

# Example of a system constraint

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

and the attacker initially knows $T_0 = \{\text{init}\}$.

One possible associated constraint system is :

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

Is there a solution ?

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

# How to solve constraint system ?

Given $\mathcal{C} = \left\{ \begin{array}{l} T_0 \Vdash u_1 \\ T_0, v_1 \Vdash u_2 \\ \quad \ldots \\ T_0, v_1, .., v_n \Vdash u_{n+1} \end{array} \right.$

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

# An easy case : "solved constraint systems"

## General case

Given $\mathcal{C} = \left\{ \begin{array}{l} T_0 \Vdash u_1 \\ T_0, v_1 \Vdash u_2 \\ \quad \ldots \\ T_0, v_1, .., v_n \Vdash u_{n+1} \end{array} \right.$

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

# An easy case : "solved constraint systems"

## General case

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

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

## Solved constraint systems

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

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

# An easy case : "solved constraint systems"

## General case

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

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

## Solved constraint systems

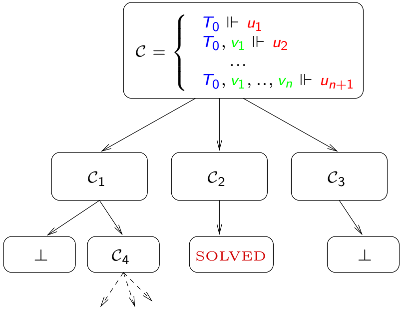
Given $\mathcal{C} = \begin{cases} T_0 \Vdash x_1 \\ T_0, v_1 \Vdash x_2 \\ \quad \dots \\ T_0, v_1, .., v_n \Vdash x_{n+1} \end{cases}$

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

Of course yes !

# Decision procedure [Millen / Comon-Lundh]

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




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

# Example : Intruder step

The intruder can built messages

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

# Example : Intruder step

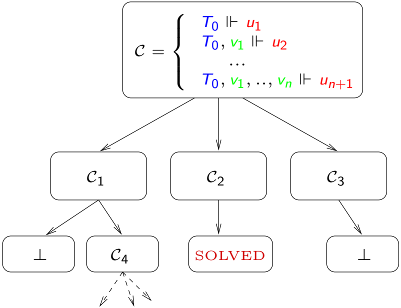
The intruder can built messages

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

Example :

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

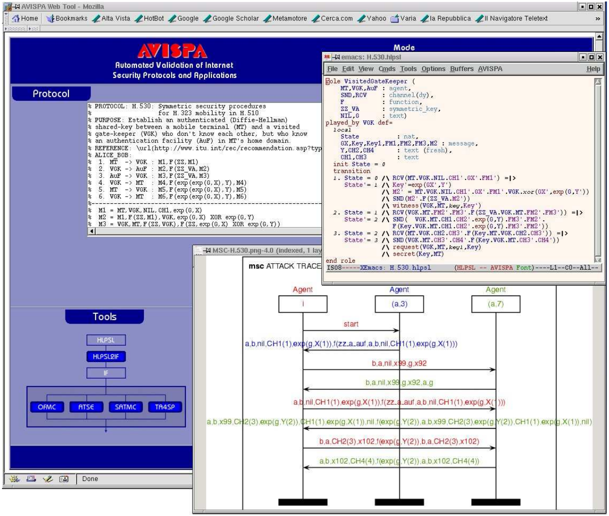
# NP-procedure for solving constraint systems




## Theorem

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

# Example of tool : Avispa Platform




www.avispa-project.org

## Collaborators

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

# 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.
  $\longrightarrow$ 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 ?

$\longrightarrow$ In general, it is undecidable !

i.e. there exists no algorithm for checking e.g. secrecy

(e.g. reduction to Post Correspondence Problem)

# How to decide security for unlimited sessions ?

→ In general, it is undecidable !

i.e. there exists no algorithm for checking e.g. secrecy

(e.g. reduction to Post Correspondence Problem)

## How to circumvent undecidability ?

- Find decidable subclasses of protocols.
- Design semi-decision procedures, that work in practice
- ...

# How to model an unbounded number of sessions ?

*"For any $x$, if the agent $A$ receives* $\mathrm{enc}(x, k_a)$ *then $A$ responds with $x$."*

$\longrightarrow$ Use of first-order logic.

# Intruder

Horn clauses perfectly reflects the attacker symbolic manipulations on terms.

| | | | |
|---|---|---|---|
| $I(x), I(y)$ | $\Rightarrow$ | $I(< x, y >)$ | pairing |
| $I(x), I(y)$ | $\Rightarrow$ | $I(\{x\}_y)$ | encryption |
| $I(\{x\}_y), I(y)$ | $\Rightarrow$ | $I(x)$ | decryption |
| $I(< x, y >)$ | $\Rightarrow$ | $I(x)$ | projection |
| $I(< x, y >)$ | $\Rightarrow$ | $I(y)$ | projection |

# Protocol

Protocol :

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

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

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

Horn clauses :

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

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

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

# Protocol

Protocol :

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

Horn clauses :

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

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, for example :

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

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

# Clauses for protocols

Intruder clauses are of the form

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

Protocol clauses

$$\begin{array}{rcl} & \Rightarrow & I(\{\mathrm{pin}\}_{k_a}) \\ I(x) & \Rightarrow & I(\{x\}_{k_b}) \\ I(\{x\}_{k_a}) & \Rightarrow & I(x) \end{array}$$

At most one variable per clause !

# Clauses for protocols

Intruder clauses are of the form

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

Protocol clauses

$$\begin{array}{rcl} & \Rightarrow & I(\{\mathrm{pin}\}_{k_a}) \\ I(x) & \Rightarrow & I(\{x\}_{k_b}) \\ I(\{x\}_{k_a}) & \Rightarrow & I(x) \end{array}$$

At most one variable per clause !

**Theorem (Hubert Comon-Lundh & VC)**

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

*Then ordered 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 [RusinowitchTuruani CSFW01]
  $\longrightarrow$ 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 [RusinowitchTuruani CSFW01]
  $\longrightarrow$ 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]

  $\longrightarrow$ some tools for proving security (ProVerif, EVA Platform)

# Going further

or Can I get my security proof for free ? (I)

- Protocols are analysed in isolation
  $\longrightarrow$ Not taking into account other protocols.
- Existing tools allow us to verify relatively small protocols
  $\longrightarrow$ They do not scale up well

# Going further

or Can I get my security proof for free ? (I)

- Protocols are analysed in isolation
  $\longrightarrow$ Not taking into account other protocols.
- Existing tools allow us to verify relatively small protocols
  $\longrightarrow$ They do not scale up well

Is it possible to compose protocols ?

# In general, no !

Protocols do not compose well as soon as they share data.

## Protocol 1

$$P_1 : \quad A \rightarrow B : \mathrm{enca}(s, \mathrm{pub}(B))$$

**Question**

Does $s$ remain confidential ?

# In general, no !

Protocols do not compose well as soon as they share data.

**Protocol 1**

$$P_1 : \quad A \rightarrow B : \mathrm{enca}(s, \mathrm{pub}(B))$$

**Protocol 2**

$$P_2 : \quad \begin{array}{l} A \rightarrow B : \mathrm{enca}(N_a, \mathrm{pub}(B)) \\ B \rightarrow A : N_a \end{array}$$

**Question**

Does $s$ remain confidential ?

# A first result : parallel composition

Joint work with Stéphanie Delaune

**Theorem**

$$\nu k\ P \models \phi \quad \Rightarrow \quad \nu k(P \mid Q) \models \phi$$

*where $\phi$ is typically a confidentiality or authentication property.*

Protocols can safely share data, provided that :

- confidential data (typically keys) do not appear in plaintext
- protocols are tagged $\longrightarrow$ reusing an idea of Joshua Guttman

# A first result : parallel composition

Joint work with Stéphanie Delaune

**Theorem**

$$\nu k\ P \models \phi \quad \Rightarrow \quad \nu k(P \mid Q) \models \phi$$

*where $\phi$ is typically a confidentiality or authentication property.*

Protocols can safely share data, provided that :

- confidential data (typically keys) do not appear in plaintext
- protocols are tagged $\longrightarrow$ reusing an idea of Joshua Guttman

Protocol $P_1$

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

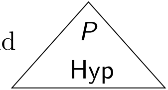
Protocol $P_2$

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

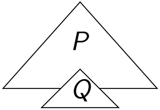
# More generally

Assume that a protocol $P \models \phi$, assuming some hypotheses Hyp on the implementation (e.g. authenticate or confidential channels).
How can we securely implement Hyp ?

$Q \models t(\mathrm{Hyp})$ and



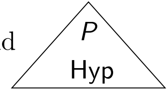

$\models \phi \quad \overset{?}{\Longrightarrow}$

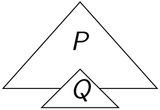



$\models \phi$

# More generally

Assume that a protocol $P \models \phi$, assuming some hypotheses Hyp on the implementation (e.g. authenticate or confidential channels).
How can we securely implement Hyp ?

$Q \models t(\mathrm{Hyp})$ and 

$\models \phi \quad \overset{?}{\Longrightarrow}$ 

$\models \phi$

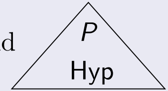
Useful for both

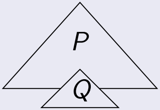
- **analysis** It is possible to analyse protocols component by component
- **design** It makes sense to design a protocol assuming some black boxes e.g. for establishing secure channels

# Modular composition

Joint work with Stefan Ciobaca

**Theorem**

$Q \models t(\mathrm{Hyp})$ and 

$\models \phi \overset{?}{\Longrightarrow}$ 

$\models \phi$

The implication holds for arbitrary composition of $P$ and $Q$ provided that :

- Shared datas between $P$ and $Q$ are secret (for each protocol, before composition)
- Primitives of $P$ and $Q$ are disjoint or tagged.

# Limitations of this approach ?

Are you ready to use any protocol verified with these techniques ?

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

# Outline of the talk

# Specificity of cryptographic models

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

$\longrightarrow$ very little abstract model

# Encryption : the old time

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

# Encryption : the old time

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

$\rightarrow$ subject to statistical analysis (Analyzing letter frequencies)

# Encryption nowadays

$\rightarrow$ Based on algorithmically hard problems.

RSA Function $n = pq$, $p$ et $q$ primes.
$e$ : public exponent

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

# Encryption nowadays

→ Based on algorithmically hard problems.

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

$e$ : public exponent

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

Diffie-Hellman Problem

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

# Encryption nowadays

→ Based on algorithmically hard problems.

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

$e$ : public exponent

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

Diffie-Hellman Problem

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

→ 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

$\longrightarrow$ Lower bound for RSA and Diffie-Hellman.

# 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).

negligible : .

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

$\eta$ : 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.

# 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 |

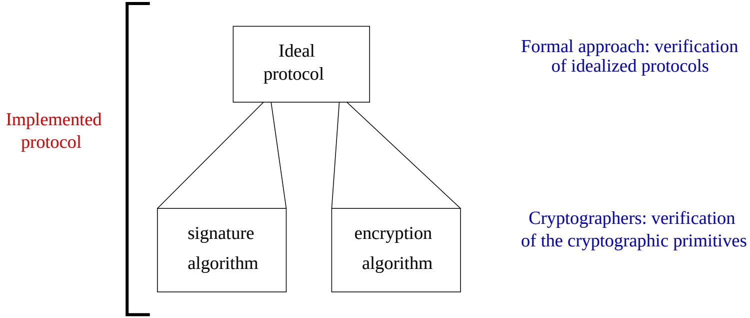
# Formal and Cryptographic approaches

| | 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 ?

# Can I get my security proof for free (II)

## Automatic cryptographically sound proofs



# Two intuitively similar definitions

**Definition (Computational indistinguishability)**

$P \approx Q$ if for any adversary $\mathcal{A}$ (that is any PPT Turing machine)
$|\Pr\{r, r'(P(r) \| \mathcal{A}(r')) = 1\}| - |\Pr\{r, r'(Q(r) \| \mathcal{A}(r')) = 1\}|$
is negligible.

Intuitively, an attacker cannot tell the difference between $P$ and $Q$.

# Two intuitively similar definitions

**Definition (Computational indistinguishability)**

$P \approx Q$ if for any adversary $\mathcal{A}$ (that is any PPT Turing machine)
$|\Pr\{r, r'(P(r)\|\mathcal{A}(r')) = 1\}| - |\Pr\{r, r'(Q(r)\|\mathcal{A}(r')) = 1\}|$
is negligible.

Intuitively, an attacker cannot tell the difference between $P$ and $Q$.

There exists a similar symbolic definition !

**Definition (observational equivalence)**

$P \sim_o Q$ if for any process $O$, we have $P\|O \sim Q\|O$.

Intuitively, an observer cannot tell the difference between $P$ and $Q$.

# Result : Soundness of observational equivalence

Joint work with Hubert Comon-Lundh

Observational equivalence is a sound abstraction of computational indistinguishability.

$$P \sim_o Q \Rightarrow [\![P]\!] \approx [\![Q]\!]$$

where $[\![P]\!]$ denotes the computational implementation of $P$.

- For simple processes
  (A fragment of applied pi-calculus that captures most security protocols)
- For IND-CCA2 symmetric encryption and pairing.
- Assuming a key hierarchy : there exists an order $<$ such that no key encrypts a smaller key.

(+ some few implementations details)

# Proof technique

## Step 1

**Lemma (Extension of [Micciancio Warinschi TCC'04])**

*Every concrete trace is the image of a valid formal trace, except with negligible probability, for symmetric encryption and pairing.*

Proof technique : Reducing the protocol security to the robustness of the primitives

# Proof technique

## Step 1

**Lemma (Extension of [Micciancio Warinschi TCC'04])**

*Every concrete trace is the image of a valid formal trace, except with negligible probability, for symmetric encryption and pairing.*

Proof technique : Reducing the protocol security to the robustness of the primitives

## Step 2

Introduction of process computation trees = generalized execution trees $T_P$.

$$P \sim_o Q \Rightarrow T_P \sim T_Q \Rightarrow T_P \approx T_Q \Rightarrow [\![P]\!] \approx [\![Q]\!]$$

# Some related work

- Abadi-Rogaway (passive attackers)

$$[M_1, \ldots, M_k] \sim [M'_1, \ldots, M'_k] \Rightarrow [\![M_1, \ldots, M_k]\!] \approx [\![M'_1, \ldots, M'_k]\!]$$

- Backes-Pfitzman et al (active attackers)
  Simulatable cryptographic library
- Canetti-Herzog (active attackers)
  Universally composable symbolic analysis
- Warinschi et al (active attackers)
  Any concrete execution is captured by a symbolic execution (except with negligible probability).

Nice, isn't it ?

But... The Devil lies in the details

# Some few implementation details

- parsing :
  - each bit-string has a label which indicates his type (identity, nonce, key, ciphertext, ...)
  - ciphertext are tagged with a label that indicates which key is used.
    Typically $k = k_1 \| k_2$ and $\mathrm{enc}(m, k) = k_1 \| \{m\}_{k_2}$.

# Some few implementation details

- parsing :
  - each bit-string has a label which indicates his type (identity, nonce, key, ciphertext, ...)
  - ciphertext are tagged with a label that indicates which key is used.
    Typically $k = k_1 \| k_2$ and $\mathrm{enc}(m, k) = k_1 \| \{m\}_{k_2}$.
- Existence of a symbolic length function
- authenticated key : the adversary can only use honestly generated keys (counter-examples otherwise).

Let's have a closer look...

# Symbolic length function

A cyphertext *a priori* reveals the length of the underlying plaintext.

$$\{n\}_k \stackrel{?}{=} \{n, n\}_k$$

Two solutions :

1) Length concealing encryption scheme :

- Requires an a priori known bound on the length of the messages (not realistic for certain protocols)
- Heavy implementation !

# Symbolic length function

A cyphertext *a priori* reveals the length of the underlying plaintext.

$$\{n\}_k \stackrel{?}{\equiv} \{n, n\}_k$$

Two solutions :

1) Length concealing encryption scheme :

2) Symbolic length function $l$ such that $l(t_1) = l(t_2)$ iff the implementation of $t_1$ and $t_2$ have the same length. Then

- $l(\langle t_1, t_2 \rangle) = l(t_1) + l(1_2) + a$
- $l(\{t_1\}_{t_2}) = l(t_1) + l(1_2) + b$

# Symbolic length function

A cyphertext *a priori* reveals the length of the underlying plaintext.

$$\{n\}_k \stackrel{?}{\equiv} \{n, n\}_k$$

Two solutions :

1) Length concealing encryption scheme :

2) Symbolic length function $l$ such that $l(t_1) = l(t_2)$ iff the implementation of $t_1$ and $t_2$ have the same length. Then

- $l(\langle t_1, t_2 \rangle) = l(t_1) + l(1_2) + a$
- $l(\{t_1\}_{t_2}) = l(t_1) + l(1_2) + b$
- $a$ and $b$ must be a multiple of the security parameter $\eta$ !
- non trivial decidability issues...

# Hidden ciphertext

$$
\begin{array}{llll}
A \rightarrow B & : & A, k, \{\{k'\}_k\}_{K_{ab}} & k, k' \text{ fresh keys} \\
B \rightarrow A & : & \{k'\}_{K_{ab}} & \\
A \rightarrow & : & \text{bad state} & \text{if } A \text{ receives } \{A\}_{K_{ab}}
\end{array}
$$

# Hidden ciphertext

$$\begin{array}{rclll} A \rightarrow & B & : & A, k, \{\{k'\}_k\}_{K_{ab}} & k, k' \text{ fresh keys} \\ B \rightarrow & A & : & \{k'\}_{K_{ab}} & \\ A \rightarrow & & : & \text{bad state} & \text{if } A \text{ receives } \{A\}_{K_{ab}} \end{array}$$

## Computational attack

The attacker can choose $k''$ such that $\mathrm{dec}(\{k'\}_k, k'') = A$, even not knowing $\{k'\}_k$.

$$\begin{array}{rcll} I \rightarrow & B & : & A, k'', \{\{k'\}_k\}_{K_{ab}} \\ B \rightarrow & A & : & \{A\}_{K_{ab}} \\ A \rightarrow & & : & \text{bad state!} \end{array}$$

# Hidden ciphertext

$$
\begin{array}{llcll}
A \rightarrow & B & : & A, k, \{\{k'\}_k\}_{K_{ab}} & k, k' \text{ fresh keys} \\
B \rightarrow & A & : & \{k'\}_{K_{ab}} & \\
A \rightarrow & & : & \text{bad state} & \text{if } A \text{ receives } \{A\}_{K_{ab}}
\end{array}
$$

## Computational attack

The attacker can choose $k''$ such that $\mathrm{dec}(\{k'\}_k, k'') = A$, even not knowing $\{k'\}_k$.

$\rightarrow$ idea : enrich again the symbolic setting ?

E.g. $\dfrac{m}{\mathrm{fakekey2}(m)}$ $\quad \mathrm{dec}(c, \mathrm{fakekey2}(m)) = m \quad$ for any $c$

# Simultaneous ciphertexts

$$
\begin{array}{llll}
A \rightarrow & B & : & c_1, \ldots, c_p \\
B \rightarrow & A & : & \{N_b, c_1, \ldots, c_p\}_{K_{ab}}, N_1, \ldots, N_p \\
A \rightarrow & B & : & k, \{N_b, c_1, \ldots, c_p\}_{K_{ab}} \\
B \rightarrow & & : & \text{bad state} \quad \text{if } B \text{ receives } k, \{N_b, \{N_1\}_k, \ldots, \{N_p\}_k\}_{K_{ab}}
\end{array}
$$

$c_1, \ldots, c_p$ ciphertexts

# Simultaneous ciphertexts

$$\begin{array}{lcl}
I \to B & : & c_1, \ldots, c_p \qquad\qquad c_1, \ldots, c_p \text{ ciphertexts} \\
B \to A & : & \{N_b, c_1, \ldots, c_p\}_{K_{ab}}, N_1, \ldots, N_p \\
I \to B & : & k', \{N_b, c_1, \ldots, c_p\}_{K_{ab}} \\
B \to & : & \text{bad state} \quad \text{if } B \text{ receives } k, \{N_b, \{N_1\}_k, \ldots, \{N_p\}_k\}_{K_{ab}}
\end{array}$$

**Computational attack**

The attacker chooses $c_1, \ldots, c_p$ and $k'$ such that $\mathrm{dec}(c_i, k') = N_b$ for all $1 \leq i \leq p$.

# Simultaneous ciphertexts

$$
\begin{array}{llll}
I \rightarrow & B & : & c_1, \ldots, c_p \qquad\qquad c_1, \ldots, c_p \text{ ciphertexts} \\
B \rightarrow & A & : & \{N_b, c_1, \ldots, c_p\}_{K_{ab}}, N_1, \ldots, N_p \\
I \rightarrow & B & : & k', \{N_b, c_1, \ldots, c_p\}_{K_{ab}} \\
B \rightarrow & & : & \text{bad state} \quad \text{if } B \text{ receives } k, \{N_b, \{N_1\}_k, \ldots, \{N_p\}_k\}_{K_{ab}}
\end{array}
$$

Computational attack

The attacker chooses $c_1, \ldots, c_p$ and $k'$ such that $\mathrm{dec}(c_i, k') = N_b$ for all $1 \leq i \leq p$.

$\rightarrow$ idea : Yet another rule ?

$$\frac{c_1 \quad \cdots \quad c_p \quad m_1 \quad \cdots \quad m_p}{\mathrm{fakekey3}(c_1, \ldots, c_p, m_1, \ldots, m_p)}$$

$$\mathrm{dec}(c_i, \mathrm{fakekey3}(c_1, \ldots, c_p, m_1, \ldots, m_p)) = m_i$$

# Playing with dishonest encryption

$$
\begin{array}{llll}
A \rightarrow & B & : & \{N_a\}_{K_{ab}} \\
C \rightarrow & B & : & k \\
B \rightarrow & A & : & k, \{\{N_a\}_k\}_{K_{ab}} \\
A \rightarrow & & : & \text{bad state}
\end{array}
$$

$c$ ciphertext

if $A$ receives $k, \{\{N_a, N_a\}_k\}_{K_{ab}}$

# Playing with dishonest encryption

$$\begin{array}{lcl} A \rightarrow B & : & \{N_a\}_{K_{ab}} \\ C \rightarrow B & : & k' \\ B \rightarrow A & : & k', \{\{N_a\}_{k'}\}_{K_{ab}} \\ A \rightarrow & : & \text{bad state} \end{array}$$

$c$ ciphertext

if $A$ receives $k, \{\{N_a, N_a\}_k\}_{K_{ab}}$

**Computational attack**

The attacker can choose $k'$ such that $\mathrm{dec}(\mathrm{enc}(N_a, k'), k') = N_a, N_a$

# Playing with dishonest encryption

$$
\begin{array}{llcl}
A \rightarrow & B & : & \{N_a\}_{K_{ab}} \\
C \rightarrow & B & : & k' \\
B \rightarrow & A & : & k', \{\{N_a\}_{k'}\}_{K_{ab}} \\
A \rightarrow & & : & \text{bad state}
\end{array}
$$

$c$ ciphertext

if $A$ receives $k, \{\{N_a, N_a\}_k\}_{K_{ab}}$
$k, \{\{N_a, N_a, N_a\}_k\}_{K_{ab}}$

**Computational attack**

The attacker can choose $k'$ such that $\mathrm{dec}(\mathrm{enc}(N_a, k'), k') = N_a, N_a$
$\mathrm{dec}(\mathrm{enc}(N_a, k'), k') = N_a, N_a, N_a$

# Playing with dishonest encryption

$$
\begin{array}{rcl}
A \rightarrow & B & : & \{N_a\}_{K_{ab}} \\
C \rightarrow & B & : & k' \\
B \rightarrow & A & : & k', \{\{N_a\}_{k'}\}_{K_{ab}} \\
A \rightarrow & & : & \text{bad state}
\end{array}
$$

$c$ ciphertext

if $A$ receives $k, \{\{N_a, N_a\}_k\}_{K_{ab}}$
$k, \{\{N_a, N_a, N_a\}_k\}_{K_{ab}}$
$k, \{\{N_a, A\}_k\}_{K_{ab}}$
$\ldots$

## Computational attack

The attacker can choose $k'$ such that $\mathrm{dec}(\mathrm{enc}(N_a, k'), k') = N_a, N_a$
$\mathrm{dec}(\mathrm{enc}(N_a, k'), k') = N_a, N_a, N_a$
$\mathrm{dec}(\mathrm{enc}(N_a, k'), k') = N_a, A$
$\ldots$

# Current solutions for dishonest keys

- [CCS 2008] Authenticated keys only.
  Requires an unrealistic infrastructure
- M. Backes current solution For any cypher-text $c$, for any dishonestly generated key $k$, $\mathrm{dec}(c, k)$ may yield any term.
- Ongoing work with Guillaume Scerri. Enrich the symbolic model, letting the adversary adding on-the-fly new equalities.

→ Same kind of issues with e.g. hash function (cf Dominique Unruh recent work)

# 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.
  $\Rightarrow$ 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
  - Equational theories (e.g. theories for e-voting protocols)
  - More complex structures for data (list, XML, ...)
  - Recursive protocols (e.g. to transform a list)
  - Proving more complex security properties like equivalence-based properties (e.g. for anonymity or e-voting protocols)
- With cryptographic guarantees
  - More primitives and security properties.
  - Is it possible to consider weaker cryptographic primitives ?
  - How far can we go ?