Verification techniques for cryptographic protocols

Véronique Cortier

RTA’08

\(^1\)LORIA, CNRS - INRIA Cassis project, Nancy Universities
Context: cryptographic protocols

- **Widely used**: web (SSH, SSL, ...), pay-per-view, electronic purse, mobile phone, ...

- **Should ensure**: confidentiality, authenticity, integrity, anonymity, ...
Context: cryptographic protocols

- **Widely used**: web (SSH, SSL, ...), pay-per-view, electronic purse, mobile phone, ...
- **Should ensure**: confidentiality, authenticity, integrity, anonymity, ...
- **Presence of an attacker**
  - may *read* every message sent on the net,
  - may *intercept and send* new messages.
Example: Credit Card Payment Protocol

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

4 actors: Bank, Customer, Card and Terminal.

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

Card owns
- Data: last name, first name, card’s number, expiration date,
- Signature’s Value $VS = \{\text{hash(Data)}\}_{K_B^{-1}}$,
- 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. $Ca \rightarrow T : Data, \{hash(Data)\}_{K_B^{-1}}$
Credit card payment Protocol (in short)

The terminal reads the card:

1. \( Ca \rightarrow T : Data, \{hash(Data)\}_{K^{-1}} \)

The terminal asks for the secret code:

2. \( T \rightarrow Cu : secret\ code? \)
3. \( Cu \rightarrow Ca : 1234 \)
4. \( Ca \rightarrow T : ok \)
Credit card payment Protocol (in short)

The terminal reads the card:

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

The terminal asks for the secret code:

2. \( \text{T} \rightarrow \text{Cu} : \text{secret code?} \)
3. \( \text{Cu} \rightarrow \text{Ca} : 1234 \)
4. \( \text{Ca} \rightarrow \text{T} : \text{ok} \)

The terminal calls the bank:

5. \( \text{T} \rightarrow \text{B} : \text{auth?} \)
6. \( \text{B} \rightarrow \text{T} : \text{N}_b \)
7. \( \text{T} \rightarrow \text{Ca} : \text{N}_b \)
8. \( \text{Ca} \rightarrow \text{T} : \{\text{N}_b\}_{K_{CB}} \)
9. \( \text{T} \rightarrow \text{B} : \{\text{N}_b\}_{K_{CB}} \)
10. \( \text{B} \rightarrow \text{T} : \text{ok} \)
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 built.

How does the “YesCard” work?

Logical flaw

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

Logical flaw

1. \( Ca \rightarrow T \) : Data, \( \{hash(Data)\}_{K^{-1}} \)
2. \( T \rightarrow Ca \) : secret code?
3. \( Cu \rightarrow Ca' \) : 2345
4. \( Ca' \rightarrow T \) : ok
How does the “YesCard” work?

Logical flaw

1. \( Ca \rightarrow T \) : 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 (Serge Humpich).
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 (Serge Humpich).

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

1. **Introduction**
   - Context
   - A famous attack

2. **Formal models**
   - Intruder
   - Protocol
   - Solving constraint systems
   - A survey of results

3. **Adding equational theories**
   - Motivation
   - Intruder problem
   - Some results

4. **Towards more guarantees**
   - Cryptographic models
   - Linking Formal and cryptographic models
   - Conclusion
Motivation: Cryptography does not suffice to ensure security!

Example: Commutative encryption (RSA)

\[ \{ \text{pin} : 3443 \}^k_{\text{alice}} \]
Motivation: Cryptography does not suffice to ensure security!

Example: Commutative encryption (RSA)

\[
\begin{align*}
\{ \text{pin} : & \, 3443 \}^{k_{\text{alice}}} \\
\{ \{ \text{pin} : 3443 \}^{k_{\text{alice}}} \}^{k_{\text{bob}}} \\
\end{align*}
\]
Motivation: Cryptography does not suffice to ensure security!

Example: Commutative encryption (RSA)

Motivation: Cryptography does not suffice to ensure security!

Example: Commutative encryption (RSA)

Since \( \{\{\text{pin} : 3443\}\}_k_{\text{alice}} \) \( k_{\text{bob}} \) = \( \{\{\text{pin} : 3443\}\}_k_{\text{bob}} \) \( k_{\text{alice}} \)
Motivation: Cryptography does not suffice to ensure security!

Example: Commutative encryption (RSA)

\[ \{ \text{pin : 3443} \}^{k_{\text{alice}}} \rightarrow \{ \{ \text{pin : 3443} \}^{k_{\text{alice}}} \}^{k_{\text{bob}}} \]

\[ \{ \text{pin : 3443} \}^{k_{\text{bob}}} \rightarrow \{ \text{pin : 3443} \}^{k_{\text{alice}}} \]

→ It does not work! (Authentication problem)
Motivation: Cryptography does not suffice to ensure security!

Example: Commutative encryption (RSA)

\[
\begin{align*}
\{ \text{pin : 3443} \}^{k_{\text{alice}}} & \quad \rightarrow \quad \{ \{ \text{pin : 3443} \}^{k_{\text{alice}}} \}^{k_{\text{bob}}} \\
\{ \text{pin : 3443} \}^{k_{\text{bob}}} & \quad \leftarrow \quad \{ \{ \text{pin : 3443} \}^{k_{\text{alice}}} \}^{k_{\text{intruder}}} \quad \rightarrow \quad \{ \text{pin : 3443} \}^{k_{\text{intruder}}} 
\end{align*}
\]

→ It does not work! (Authentication problem)
Messages are abstracted by terms.

Agents: \(a, b, \ldots\)  
Nonces: \(n_1, n_2, \ldots\)  
Keys: \(k_1, k_2, \ldots\)

Cyphertext: \(\{m\}_k\)  
Concatenation: \(\langle m_1, m_2 \rangle\)

Example: The message \(\{A, Na\}_K\) is represented by:

```
\[
\begin{array}{c}
\{\} \\
\langle \rangle \\
A \\
Na
\end{array}
\]
```
Intruder abilities

Composition rules

\[
\begin{align*}
T \vdash u & \quad T \vdash v \\
\hline
T \vdash \langle u, v \rangle
\end{align*}
\]

\[
\begin{align*}
T \vdash u & \quad T \vdash v \\
\hline
T \vdash \text{enc}(u, v)
\end{align*}
\]

\[
\begin{align*}
T \vdash u & \quad T \vdash v \\
\hline
T \vdash \text{enca}(u, v)
\end{align*}
\]
Intruder abilities

Composition rules

\[
\begin{align*}
T \vdash u & \quad T \vdash v \\
\quad & \quad \Downarrow \\
T \vdash \langle u, v \rangle
\end{align*}
\]

\[
\begin{align*}
T \vdash u & \quad T \vdash v \\
\quad & \quad \Downarrow \\
T \vdash \text{enc}(u, v)
\end{align*}
\]

\[
\begin{align*}
T \vdash u & \quad T \vdash v \\
\quad & \quad \Downarrow \\
T \vdash \text{enca}(u, v)
\end{align*}
\]

Decomposition rules

\[
\begin{align*}
T \vdash u & \quad u \in T \\
\quad & \quad \Downarrow \\
T \vdash u
\end{align*}
\]

\[
\begin{align*}
T \vdash \langle u, v \rangle & \quad T \vdash v \\
\quad & \quad \Downarrow \\
T \vdash u
\end{align*}
\]

\[
\begin{align*}
T \vdash \langle u, v \rangle & \quad T \vdash v \\
\quad & \quad \Downarrow \\
T \vdash \text{enca}(u, \text{pub}(v))
\end{align*}
\]

\[
\begin{align*}
T \vdash \text{priv}(v) & \quad T \vdash u
\end{align*}
\]
Intruder abilities

Composition rules

\[
\begin{align*}
T \vdash u & \quad T \vdash v \\
\hline
T \vdash \langle u, v \rangle
\end{align*}
\]

\[
\begin{align*}
T \vdash u & \quad T \vdash v \\
\hline
T \vdash \text{enc}(u, v)
\end{align*}
\]

\[
\begin{align*}
T \vdash u & \quad T \vdash v \\
\hline
T \vdash \text{enca}(u, v)
\end{align*}
\]

Decomposition rules

\[
\begin{align*}
u \in T \\
\hline
T \vdash u
\end{align*}
\]

\[
\begin{align*}
T \vdash \langle u, v \rangle \\
\hline
T \vdash u
\end{align*}
\]

\[
\begin{align*}
T \vdash \langle u, v \rangle \\
\hline
T \vdash v
\end{align*}
\]

\[
\begin{align*}
T \vdash \text{enc}(u, v) & \quad T \vdash v \\
\hline
T \vdash u
\end{align*}
\]

\[
\begin{align*}
T \vdash \text{enca}(u, \text{pub}(v)) & \quad T \vdash \text{priv}(v) \\
\hline
T \vdash u
\end{align*}
\]

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

\[ \langle \text{Bob, } k \rangle \rightarrow \langle \text{Alice, } \text{enc}(s, k) \rangle \]
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

\[ \langle \text{Bob, k} \rangle \]

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

\[ \text{enc(s, k)} \]

\[ \text{s} \]

Answer: Of course, Yes!

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

\[ \langle \text{Bob, k} \rangle \]

\[ \text{k} \]

\[ \text{s} \]
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
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 $m$. 

Véronique Cortier
Protocol description

Protocol:

\[
\begin{align*}
A \rightarrow B & : \quad \{\text{pin}\}_{k_a} \\
B \rightarrow A & : \quad \{\{\text{pin}\}_{k_a}\}_{k_b} \\
A \rightarrow B & : \quad \{\text{pin}\}_{k_b}
\end{align*}
\]

A protocol is a finite set of roles:

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

\[
\begin{align*}
\text{init} & \rightarrow^{k_a} \text{enc}(\text{pin}, k_a) \\
\text{enc}(x, k_a) & \rightarrow x.
\end{align*}
\]
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 1\(^{st} \) participant played by \( a \) talking to \( b \):
  
  \[
  \begin{align*}
  \text{init} & \xrightarrow{k_a} \text{enc}(\text{pin}, k_a) \\
  \text{enc}(x, k_a) & \xrightarrow{} x.
  \end{align*}
  \]

- role \( \Pi(2) \) corresponding to the 2\(^{nd} \) participant played by \( b \) with \( a \):
  
  \[
  \begin{align*}
  x & \xrightarrow{k_b} \text{enc}(x, k_b) \\
  \text{enc}(y, k_b) & \xrightarrow{} \text{stop}.
  \end{align*}
  \]
Secrecy via constraint solving

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

\[
\begin{align*}
\text{Scenario} & : \\
\text{rcv}(u_1) & \xrightarrow{N_1} \text{snd}(v_1) \\
\text{rcv}(u_2) & \xrightarrow{N_2} \text{snd}(v_2) \\
& \ldots \\
\text{rcv}(u_n) & \xrightarrow{N_n} \text{snd}(v_n)
\end{align*}
\]

\[
\text{Constraint System} \quad C = \begin{cases} 
T_0 \vdash u_1 \\
T_0, v_1 \vdash u_2 \\
& \ldots \\
T_0, v_1, \ldots, v_n \vdash s
\end{cases}
\]

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

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

**Scenario**

\[
\begin{align*}
\text{rcv}(u_1) & \xrightarrow{N_1} \text{snd}(v_1) \\
\text{rcv}(u_2) & \xrightarrow{N_2} \text{snd}(v_2) \\
\cdots \\
\text{rcv}(u_n) & \xrightarrow{N_n} \text{snd}(v_n)
\end{align*}
\]

**Constraint System**

\[
C = \left\{ \begin{array}{l}
T_0 \not\vdash u_1 \\
T_0, v_1 \not\vdash u_2 \\
\cdots \\
T_0, v_1, \ldots, v_n \not\vdash s
\end{array} \right. 
\]

**Solution of a constraint system**

A substitution \( \sigma \) such that

\[
\text{for every } T \not\vdash u \in C, u\sigma \text{ is deductible from } T\sigma, \text{ that is } u\sigma \vdash T\sigma.
\]
How to solve constraint system?

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

Question: Is there a solution $\sigma$ of $C$?
How to solve constraint system?

Given \( C = \begin{cases} \quad T_0 \models u_1 \\
\quad T_0, v_1 \models u_2 \\
\quad \ldots \\
\quad T_0, v_1, \ldots, v_n \models u_{n+1} \end{cases} \)

Question Is there a solution \( \sigma \) of \( C \)?

Advertisement:
Lecture of Hubert Comon-Lundh at ISR 2008 next week
An easy case: “solved constraint systems”

Given \( C = \begin{cases} T_0 \models x_1 \\ T_0, v_1 \models x_2 \\ \vdots \\ T_0, v_1, \ldots, v_n \models x_{n+1} \end{cases} \)

Question Is there a solution \( \sigma \) of \( C \)?
An easy case: "solved constraint systems"

Given: $C = \{ T_0 \models x_1, T_0, v_1 \models x_2, \ldots, T_0, v_1, \ldots, v_n \models x_{n+1} \}$

Question: Is there a solution $\sigma$ of $C$?

Of course yes!
Consider e.g. $\sigma(x_1) = \cdots = \sigma(x_{n+1}) = t \in T_0$. 
Goal: Transformation of the constraints in order to obtain a solved constraint system.

$$C = \left\{ T_0 \models u_1, T_0, v_1 \models u_2, \ldots, T_0, v_1, \ldots, v_n \models u_{n+1} \right\}$$

$C$ has a solution iff $C \leadsto C'$ with $C'$ in solved form.
The intruder can build messages

\[ R_5 : \quad C \land T \not\models f(u, v) \leadsto C \land T \not\models u \land T \not\models v \]

for \( f \in \{\langle\rangle, \text{enc}, \text{enca}\} \)
The intruder can build messages

\[ R_5 : \quad C \land T \not\models f(u, v) \leadsto C \land T \not\models u \land T \not\models v \]
for \( f \in \{\langle \rangle, \text{enc}, \text{enca}\} \)

Example:

\[ a, k \not\models \text{enc}(\langle x, y \rangle, k) \leadsto a, k \not\models x \]
\[ a, k \not\models y \]
The constraint $\text{enc}(s, x) \models s$ will be satisfied as soon as $k \models x$ is satisfied.
Eliminating redundancies

\[ k \models x \]
\[ \text{enc}(s, x) \models s \]

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

\[ R_1 : \mathcal{C} \land T \models u \rightsquigarrow \mathcal{C} \quad \text{if} \quad T \cup \{x \mid T' \models x \in \mathcal{C}, T' \subset T\} \models u \]
Unsolvable constraints

\[ R_4 : C \land T \not\mathcal{F} u \rightsquigarrow \bot \quad \text{if } \text{var}(T, u) = \emptyset \text{ and } T \not\models u \]

Example:

\[ \cdots \]

\[ a, \text{enc}(s, k) \not\mathcal{F} s \rightsquigarrow \bot \]

\[ \cdots \]
Guessing equalities

Example: \( k, \text{enc}(\text{enc}(x, k'), k) \models \text{enc}(a, k') \)

\[
R_2 : C \land T \models u \xrightarrow{\sigma} C\sigma \land T\sigma \models u\sigma \quad u' \in \text{st}(T) \\
\text{if } \sigma = \text{mgu}(u, u'), \ u, u' \notin \mathcal{X}, \ u \neq u'
\]
Guessing equalities

1. Example: $k, \text{enc(\text{enc}(x, k'), k)} \not\vdash \text{enc}(a, k')$

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

2. Example: $\text{enc}(s, \langle a, x \rangle), \text{enc}(\langle y, b \rangle, k), k \not\vdash s$

$$R_3 : \mathcal{C} \land T \not\vdash v \rightsquigarrow_{\sigma} C\sigma \land T\sigma \not\vdash v\sigma \quad u, u' \in \text{st}(T)$$
if $\sigma = \text{mgu}(u, u')$, $u, u' \notin \mathcal{X}$, $u \neq u'$
**Theorem**

- $C$ has a solution iff $C \leadsto C'$ with $C'$ in solved form.
- $\leadsto$ is terminating in polynomial time.
What formal methods allow to do?

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

For a bounded number of sessions, secrecy is co-NP-complete [RusinowitchTuruani CSFW01]
→ numerous 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**
  
  \[\text{RusinowitchTuruani CSFW01}\]

  \(\rightarrow\) numerous tools for detecting attacks (Casper, Avispa platform...)

- For an unbounded number of sessions
  
  - for **one-copy protocols**, secrecy is **DEXPTIME-complete**
    
    \[\text{CortierComon RTA03} \ [\text{SeildVerma LPAR04}]\]
  
  - for **message-length bounded protocols**, secrecy is **DEXPTIME-complete**
    
    \[\text{Durgin et al FMSP99} \ [\text{Chevalier et al CSL03}]\]

  \(\rightarrow\) some tools for proving security (ProVerif, EVA Platform)
Many tools for a bounded number of sessions (search for attacks): Casper, Avispa platform, ...

Some tools for an unbounded number of sessions (security proof): ProVerif, EVA platform

- new attacks have been discovered (e.g. the man-in-the-middle attack on the Needham-Schroeder protocol)
- hundreds protocols analyzed in few minutes or few seconds for most of them
- real-world applications (IETF, ...)
Example of tool: Avispa Platform

Collaborators
- Cassis project, Loria
- DIST, Italy
- ETHZ, Swiss
- Siemens, Germany

www.avispa-project.org

Véronique Cortier
Verification techniques for cryptographic protocols
Outline of the talk

1. Introduction
   - Context
   - A famous attack

2. Formal models
   - Intruder
   - Protocol
   - Solving constraint systems
   - A survey of results

3. Adding equational theories
   - Motivation
   - Intruder problem
   - Some results

4. Towards more guarantees
   - Cryptographic models
   - Linking Formal and cryptographic models
   - Conclusion
Back to our running example:

\[
\begin{align*}
A \rightarrow B &: \{\text{pin}\}_{k_a} \\
B \rightarrow A &: \{\{\text{pin}\}_{k_a}\}_{k_b} \\
A \rightarrow B &: \{\text{pin}\}_{k_b}
\end{align*}
\]

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

\[ x \oplus (y \oplus z) = z \quad x \oplus y = y \oplus x \quad x \oplus x = 0 \quad x \oplus 0 = x \]

Diffie-Hellmann

\[ \exp(\exp(z, x), y) = \exp(\exp(z, y), x) \]
E-voting protocols

First phase:

\[ V \rightarrow A : \text{sign}(\text{blind}(\text{vote}, r), V) \]
\[ A \rightarrow V : \text{sign}(\text{blind}(\text{vote}, r), A) \]

Voting phase:

\[ V \rightarrow C : \text{sign}(\text{vote}, A) \]

...
Equational theory for blind signatures

[Kremer Ryan 05]

\[
\begin{align*}
\text{checksign}(\text{sign}(x, y), \text{pk}(y)) &= x \\
\text{unblind}(\text{blind}(x, y), y) &= x \\
\text{unblind}(\text{sign}(\text{blind}(x, y), z), y) &= \text{sign}(x, z)
\end{align*}
\]
Deduction

\[
\frac{T \vdash_E M}{M \in T}
\]

\[
\frac{T \vdash_E M_1 \quad \cdots \quad T \vdash_E M_k}{T \vdash_E f(M_1, \ldots, M_k) \quad f \in \Sigma}
\]

\[
\frac{T \vdash M}{M =_E M'}
\]
Deduction

\[
\frac{M \in T}{T \vdash_E M} \quad \quad \quad \quad \quad \frac{T \vdash_E M_1 \quad \cdots \quad T \vdash_E M_k}{T \vdash_E f(M_1, \ldots, M_k)} \quad f \in \Sigma
\]

\[
\frac{T \vdash M}{T \vdash M'} \quad \quad \quad \quad \quad M =_E M'
\]

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

\[
\frac{T \vdash \text{enc}(\text{secret}, k) \quad T \vdash k}{T \vdash \text{dec}(\text{enc}(\text{secret}, k), k)} \quad f \in \Sigma
\]

\[
\frac{T \vdash \text{secret}}{T \vdash \text{dec}(\text{enc}(x, y), y) = x}
\]

Véronique Cortier Verification techniques for cryptographic protocols
For analyzing equational theories, we (try to) associate to $E$ a finite convergent rewriting system $\mathcal{R}$ such that:

$$u =_E v \iff u \downarrow = v \downarrow$$

**Definition (Characterization of the deduction relation)**

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

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

(Also called Cap Intruder problem [Narendran et al])
## Some results with equational theories

<table>
<thead>
<tr>
<th>Security problem</th>
<th>Bounded number of sessions</th>
<th>Unbounded number of sessions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>co-NP-complete [CKRT04]</td>
<td>Ping-pong protocols: co-NP-complete [Turuani04]</td>
</tr>
<tr>
<td>Commutative encryption</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exclusive Or</td>
<td>Decidable [CS03, CKRT03]</td>
<td>One copy - No nonces: Decidable [CLC03]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Two-way automata - No nonces: Decidable [Verma03]</td>
</tr>
<tr>
<td>Abelian Groups</td>
<td>Decidable [Shmatikov04]</td>
<td>Two-way automata - No nonces: Decidable [Verma03]</td>
</tr>
<tr>
<td>Prefix encryption</td>
<td>co-NP-complete [CKRT03]</td>
<td>AC properties of the Modular Exponentiation</td>
</tr>
<tr>
<td>Abelian Groups and Modular Exponentiation</td>
<td>General case: Decidable [Shmatikov04]</td>
<td>No nonces: Semi-Decision Procedure [GLRV04]</td>
</tr>
<tr>
<td></td>
<td>Restricted protocols: co-NP-complete [CKRT03]</td>
<td></td>
</tr>
</tbody>
</table>
Outline of the talk

1. Introduction
   - Context
   - A famous attack

2. Formal models
   - Intruder
   - Protocol
   - Solving constraint systems
   - A survey of results

3. Adding equational theories
   - Motivation
   - Intruder problem
   - Some results

4. Towards more guarantees
   - Cryptographic models
   - Linking Formal and cryptographic models
   - Conclusion
Specificity of cryptographic models

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

→ very little abstract model
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 $d = e^{-1} \mod \phi(n)$
Encryption nowadays

→ Based on algorithmically hard problems.

**RSA Function** \( n = pq, p \) et \( q \) primes.

\[ e : \text{public exponent} \]

- \( x \mapsto x^e \mod n \) 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 \( \text{DH}(A, B) = g^{ab} \)
Encryption nowadays

→ Based on algorithmically hard problems.

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

- $e : \text{public exponent}$
  - $x \mapsto x^e \mod n$ 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 $\text{DH}(A, B) = g^{ab}$

→ Based on hardness of integer factorization.
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
→ Several notions of secrecy:

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

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

$\eta$: 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.
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

<table>
<thead>
<tr>
<th></th>
<th>Formal approach</th>
<th>Cryptographic approach</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Messages</strong></td>
<td>terms</td>
<td>bitstrings</td>
</tr>
<tr>
<td><strong>Encryption</strong></td>
<td>idealized</td>
<td>algorithm</td>
</tr>
<tr>
<td><strong>Adversary</strong></td>
<td>idealized</td>
<td>any polynomial algorithm</td>
</tr>
<tr>
<td><strong>Secrecy property</strong></td>
<td>reachability-based property</td>
<td>indistinguishability</td>
</tr>
<tr>
<td><strong>Guarantees</strong></td>
<td>unclear</td>
<td>strong</td>
</tr>
<tr>
<td><strong>Protocol</strong></td>
<td>complex, several sessions</td>
<td>simple, one session</td>
</tr>
</tbody>
</table>
## Formal and Cryptographic approaches

<table>
<thead>
<tr>
<th></th>
<th>Formal approach</th>
<th>Cryptographic approach</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Messages</strong></td>
<td>terms</td>
<td>bitstrings</td>
</tr>
<tr>
<td><strong>Encryption</strong></td>
<td>idealized</td>
<td>algorithm</td>
</tr>
<tr>
<td><strong>Adversary</strong></td>
<td>idealized</td>
<td>any polynomial algorithm</td>
</tr>
<tr>
<td><strong>Secrecy property</strong></td>
<td>reachability-based</td>
<td>indistinguishability</td>
</tr>
<tr>
<td></td>
<td>property</td>
<td></td>
</tr>
<tr>
<td><strong>Guarantees</strong></td>
<td>unclear</td>
<td>strong</td>
</tr>
<tr>
<td><strong>Protocol</strong></td>
<td>complex, several</td>
<td>simple, one session</td>
</tr>
<tr>
<td></td>
<td>sessions</td>
<td></td>
</tr>
<tr>
<td><strong>Proof</strong></td>
<td>automatic</td>
<td>by hand, tedious and error-prone</td>
</tr>
</tbody>
</table>

### Link between the two approaches?

Véronique Cortier

Verification techniques for cryptographic protocols
Composition of the two approaches

Automatic cryptographically sound proofs

→ Currently implemented in the Avispa platform.
**Example: correspondence of secrecy properties**

<table>
<thead>
<tr>
<th>Theorem</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Symbolic secrecy implies cryptographic indistinguishability.</strong></td>
</tr>
</tbody>
</table>

- 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),
Formal methods, including of course rewriting techniques, form a very powerful approach for analyzing security protocols

- Many automatic tools (ProVerif, Avispa, ...)
- Cryptographic guarantees
Conclusion

Formal methods, including of course rewriting techniques, form a very powerful approach for analyzing security protocols

- Many automatic tools (ProVerif, Avispa, ...)
- Cryptographic guarantees

Some current directions of research:

- Considering more equational theories (e.g. theories for e-voting protocols)
- Combining formal and cryptographic models
- Adding more complex structures for data (list, XML, ...)
- ...