How to prove the security of communication protocols?

Véronique Cortier, LORIA - CNRS, Nancy

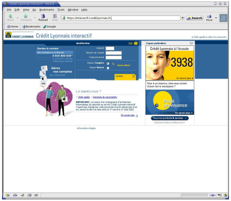
Seminar at Collège de France, 18 mai 2011

Joint work with Hubert Comon-Lundh, Stéphanie Delaune, Steve Kremer, Ben Smyth and Bogdan Warinschi.

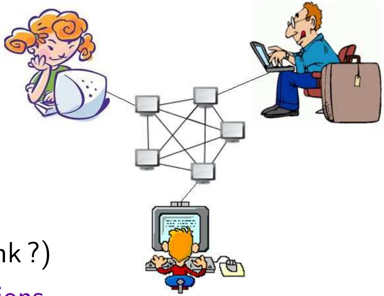
Context : cryptographic protocols

Cryptographic protocols are widely used in everyday life.

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



Security goals



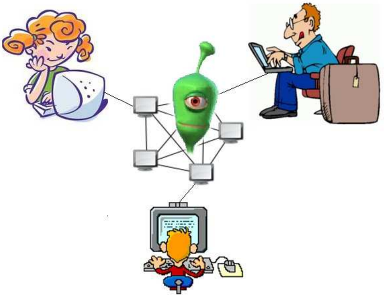
Cryptographic protocols aim at

- 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 !)

Difficulty : there are potential powerful attackers !

Presence of an attacker

- may participate to the protocol.
- may forge and send messages,
- may read every message sent on the net,
- may intercept messages,



Attacking Single Sign On Protocol

Single Sign On Protocols

- enables to log in once for several services
- used e.g. in Google App

→ A flaw discovered in 2010, now fixed (Avantssar project)

Step 1 An attacker offers an interesting or funny (but malicious) new Google App

Step 2 Some clients register to this malicious Application

Step 3 The attacker can now access all the other applications of the client, including e.g. Gmail or Google Calendar.

Designing protocols is error prone

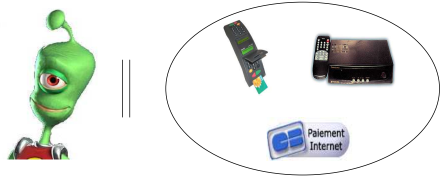
Software testing leaves flaws : *cf* Lectures of Martín Abadi

- Flaw in the authentication protocol used in Google Apps
- Attack on pay-per-view devices
- Man-in-the-middle attack

These flaws rely on the design of the protocols

- Not on a bad implementation (bugs)
- Not on weaknesses of the primitives (e.g. encryption, signatures)
- Not on generic hacking techniques (e.g. worms, code injection)

How to analyse security protocols ?



$\stackrel{?}{\models}$

confidentiality
authenticity
anonymity
non-repudiation
...

Methodology

1. Proposing accurate models
   - symbolic models
   - cryptographic/computational models
2. Proving security
   - decision procedures
   - transfer results

Running example : electronic voting

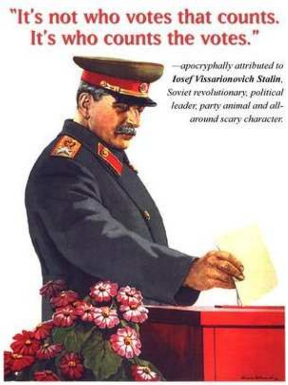
Example : Electronic voting

Elections are a security-sensitive process which is the cornerstone of modern democracy.

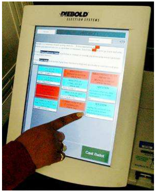
Electronic voting promises

- Convenient, efficient and secure facility for recording and tallying votes
- for a variety of types of elections : from small committees or on-line communities through to full-scale national elections

Already used e.g. in Estonia, Norway, USA.

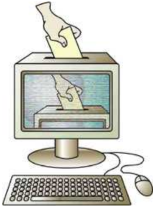


Two main families for e-voting



Voting machines

- Voters have to attend a voting station
- External authentication system (e.g. ID card)



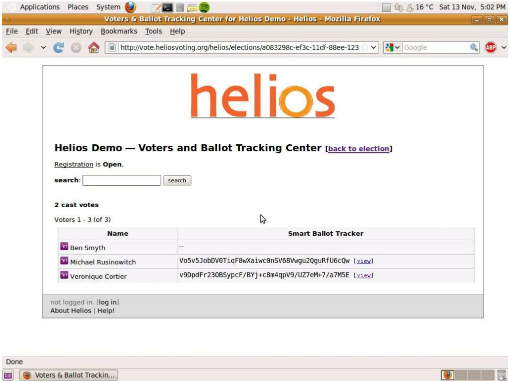
Internet voting

- Voters vote from home
- from their own computers
- Systems in use : Civitas (A. Myers *et al*), Helios, ...

cf Seminar of Ron Rivest (March 23rd).

Running example : Helios

http ://heliosvoting.org/



Developed by B. Adida *et al*, already in use :

- Election at Louvain University Princeton
- Election of the IACR board (major association in Cryptography)

Behavior of Helios (simplified)

Phase 1 : voting

Bulletin Board		
Alice	$\{v_A\}_{\mathrm{pk}(S)}$	$v_A = 0$ or 1
Bob	$\{v_B\}_{\mathrm{pk}(S)}$	$v_B = 0$ or 1
Chris	$\{v_C\}_{\mathrm{pk}(S)}$	$v_C = 0$ or 1

$\mathrm{pk}(S)$: public key, the private key being shared among trustees.

Behavior of Helios (simplified)

Phase 1 : voting

$\{v_D\}_{\mathrm{pk}(S)}$ →

Bulletin Board

Alice	$\{v_A\}_{\mathrm{pk}(S)}$	$v_A = 0$ or 1
Bob	$\{v_B\}_{\mathrm{pk}(S)}$	$v_B = 0$ or 1
Chris	$\{v_C\}_{\mathrm{pk}(S)}$	$v_C = 0$ or 1

$\mathrm{pk}(S)$: public key, the private key being shared among trustees.

Behavior of Helios (simplified)

Phase 1 : voting

Bulletin Board	
Alice	$\{v_A\}_{\mathsf{pk}(S)}$
Bob	$\{v_B\}_{\mathsf{pk}(S)}$
Chris	$\{v_C\}_{\mathsf{pk}(S)}$
David	$\{v_D\}_{\mathsf{pk}(S)}$

$v_A = 0$ or 1
$v_B = 0$ or 1
$v_C = 0$ or 1
$v_D = 0$ or 1

$\mathsf{pk}(S)$: public key, the private key being shared among trustees.

Behavior of Helios (simplified)

Phase 1 : voting

Bulletin Board

Alice	$\{v_A\}_{\mathsf{pk}(S)}$	$v_A = 0$ or 1
Bob	$\{v_B\}_{\mathsf{pk}(S)}$	$v_B = 0$ or 1
Chris	$\{v_C\}_{\mathsf{pk}(S)}$	$v_C = 0$ or 1
David	$\{v_D\}_{\mathsf{pk}(S)}$	$v_D = 0$ or 1
...	...	

Phase 2 : Tallying using homomorphic encryption (El Gamal)

$$\prod_{i=1}^{n} \{v_i\}_{\mathsf{pk}(S)} = \{\sum_{i=1}^{n} v_i\}_{\mathsf{pk}(S)}$$ based on $g^a * g^b = g^{a+b}$

$\longrightarrow$ Only the final result needs to be decrypted !

$\mathsf{pk}(S)$: public key, the private key being shared among trustees.

This is oversimplified !

$\xrightarrow{\{v_D\}_{pk(S)}}$

Bulletin Board	
Alice	$\{v_A\}_{pk(S)}$
Bob	$\{v_B\}_{pk(S)}$
Chris	$\{v_C\}_{pk(S)}$
David	$\{v_D\}_{pk(S)}$
...	...

$v_A = 0$ or 1
$v_B = 0$ or 1
$v_C = 0$ or 1

Result : $\{v_A + v_B + v_C + v_D + \cdots\}_{pk(S)}$

This is oversimplified !

$\{v_D\}_{pk(S)}$ $\longrightarrow$

Bulletin Board

Alice	$\{v_A\}_{pk(S)}$	$v_A = 0$ or 1
Bob	$\{v_B\}_{pk(S)}$	$v_B = 0$ or 1
Chris	$\{v_C\}_{pk(S)}$	$v_C = 0$ or 1
David	$\{v_D\}_{pk(S)}$	$v_D = 100$
...	...	

Result : $\{v_A + v_B + v_C + 100 + \cdots\}_{pk(S)}$

A malicious voter can cheat !

This is oversimplified !

$\xrightarrow{\{v_D\}_{\mathrm{pk}(S)}}$

Bulletin Board

Alice	$\{v_A\}_{\mathrm{pk}(S)}$	$v_A = 0 \text{ or } 1$
Bob	$\{v_B\}_{\mathrm{pk}(S)}$	$v_B = 0 \text{ or } 1$
Chris	$\{v_C\}_{\mathrm{pk}(S)}$	$v_C = 0 \text{ or } 1$
David	$\{v_D\}_{\mathrm{pk}(S)}$	$\cancel{v_D = 100}$
...	...	

Result : $\{v_A + v_B + v_C + v_D + \cdots\}_{\mathrm{pk}(S)}$

~~A malicious voter can cheat !~~

In Helios : use of (Signature of) Proof of Knowledge

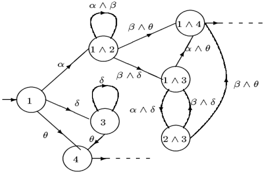
$$\{v_D\}_{\mathrm{pk}(S)}, \mathrm{SPK}\{v_D = 0 \text{ or } 1\}$$

How to analyse security protocols ?

For example, how to prove that Helios is secure ?

How to analyse security protocols ?

For example, how to prove that Helios is secure ?



Task 1 : Modeling

1. Modeling messages
2. Modeling the behavior of the protocol
3. Modeling “security”

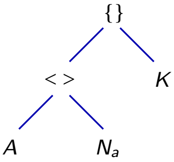
Modeling messages

Idea 1 : keeping only the structure of the messages

$\rightarrow$Messages are abstracted by terms.

Example :

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



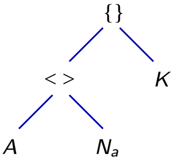
Modeling messages

Idea 1 : keeping only the structure of the messages

$\rightarrow$Messages are abstracted by terms.

Example :

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



Idea 2 : Equations for reflecting the properties of the primitives

Decryption	$\mathrm{dec}(\{x\}_y, y) = x$
Homomorphic encryption	$\{x_1\}_y * \{x_2\}_y = \{x_1 + x_2\}_y$

Modeling protocols

Processes of the applied pi-calculus, introduced by Martín Abadi

- Voter id voting v

$$\mathrm{Voter}(\mathrm{id}, \mathrm{v}) = \overline{c_{\mathrm{id}}}(\{v\}_{\mathrm{pk}(S)}, \mathrm{spk}(v, \{v\}_{\mathrm{pk}(S)}))$$

Modeling protocols

Processes of the applied pi-calculus, introduced by Martín Abadi

- Voter id voting v

$$\mathrm{Voter}(\mathrm{id}, \mathrm{v}) = \overline{c_{\mathrm{id}}}(\{v\}_{\mathrm{pk}(S)}, \mathrm{spk}(v, \{v\}_{\mathrm{pk}(S)}))$$

- Bulletin board for n voters

$$\begin{array}{lcl} \mathrm{BulletinBoard} & = & c_{\mathrm{id}_1}(x_1).\ \text{if } \mathit{Valid}(x_1) \text{ then } \overline{\mathit{out}}(x_1). \\ & & \dots \\ & & c_{\mathrm{id}_n}(x_n).\ \text{if } \mathit{Valid}(x_n) \text{ then } \overline{\mathit{out}}(x_n). \\ & & \overline{c_{tally}}(\pi_1(x_1) * \cdots * \pi_1(x_n)) \end{array}$$

Modeling protocols

Processes of the applied pi-calculus, introduced by Martín Abadi

- Voter id voting v

$$\mathrm{Voter}(\mathrm{id}, \mathrm{v}) = \overline{c_{\mathrm{id}}}(\{v\}_{\mathrm{pk}(S)}, \mathrm{spk}(v, \{v\}_{\mathrm{pk}(S)}))$$

- Bulletin board for n voters

$$\begin{array}{rcl} \mathrm{BulletinBoard} & = & c_{\mathrm{id}_1}(x_1).\ \mathrm{if}\ \mathit{Valid}(x_1)\ \mathrm{then}\ \overline{\mathit{out}}(x_1). \\ & & \ldots \\ & & c_{\mathrm{id}_n}(x_n).\ \mathrm{if}\ \mathit{Valid}(x_n)\ \mathrm{then}\ \overline{\mathit{out}}(x_n). \\ & & \overline{c_{\mathit{tally}}}(\pi_1(x_1) * \cdots * \pi_1(x_n)) \end{array}$$

- Tallying phase

$$\mathrm{Tally} = c_{\mathit{tally}}(y).\overline{\mathit{out}}(\mathrm{dec}(y, \mathrm{sk}(S)))$$

Modeling attackers

We assume that the network can be controlled by attackers

- may participate to the protocol.
- may forge and send messages,
- may read every message sent on the net,
- may intercept messages,

Modeling attackers

We assume that the network can be controlled by attackers

- may participate to the protocol.
- may forge and send messages,
- may read every message sent on the net,
- may intercept messages,

Attackers in applied pi-calculus

A protocol P satisfies some property ϕ if for all process A

$$A \mid P \models \phi$$

What is a secure voting protocol ?

Let's have a closer look to privacy

How to state formally :

"No one should know my vote (0 or 1)" ?

Idea 1 : An attacker should not learn the value of my vote.

Let's have a closer look to privacy

How to state formally :

"No one should know my vote (0 or 1)" ?

Idea 1 : An attacker should not learn the value of my vote.
But everyone knows 0 and 1 !

Let's have a closer look to privacy

How to state formally :

"No one should know my vote (0 or 1)" ?

~~Idea 1 : An attacker should not learn the value of my vote.~~

Idea 2 : An attacker should not attach my vote to my identity.

Let's have a closer look to privacy

How to state formally :

"No one should know my vote (0 or 1)" ?

~~Idea 1 : An attacker should not learn the value of my vote.~~

Idea 2 : An attacker should not attach my vote to my identity.
But everyone can form $\langle Alice, 0\rangle$ and $\langle Alice, 1\rangle$!

Let's have a closer look to privacy

How to state formally :

"No one should know my vote (0 or 1)" ?

~~Idea 1 : An attacker should not learn the value of my vote.~~

~~Idea 2 : An attacker should not attach my vote to my identity.~~

Idea 3 : An attacker cannot see the difference when I vote 0 or 1.

$$\mathrm{Voter}_1(0) \mid \mathrm{Voter}_2(v_2) \mid \cdots \mid \mathrm{Voter}_n(v_n) \sim \mathrm{Voter}_1(1) \mid \mathrm{Voter}_2(v_2) \mid \cdots \mid \mathrm{Voter}_n(v_n)$$

Let's have a closer look to privacy

How to state formally :

"No one should know my vote (0 or 1)" ?

~~Idea 1 : An attacker should not learn the value of my vote.~~

~~Idea 2 : An attacker should not attach my vote to my identity.~~

Idea 3 : An attacker cannot see the difference when I vote 0 or 1.

$$\mathrm{Voter}_1(0) \mid \mathrm{Voter}_2(v_2) \mid \cdots \mid \mathrm{Voter}_n(v_n) \sim \mathrm{Voter}_1(1) \mid \mathrm{Voter}_2(v_2) \mid \cdots \mid \mathrm{Voter}_n(v_n)$$

- The attacker always sees the difference since the tally differs.
- Unanimity does break privacy.

Let's have a closer look to privacy

How to state formally :

"No one should know my vote (0 or 1)" ?

~~Idea 1 : An attacker should not learn the value of my vote.~~

~~Idea 2 : An attacker should not attach my vote to my identity.~~

~~Idea 3 : An attacker cannot see the difference when I vote 0 or 1.~~

$$\mathrm{Voter}_1(0) \mid \mathrm{Voter}_2(v_2) \mid \cdots \mid \mathrm{Voter}_n(v_n) \sim \mathrm{Voter}_1(1) \mid \mathrm{Voter}_2(v_2) \mid \cdots \mid \mathrm{Voter}_n(v_n)$$

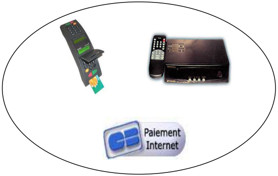
Idea 4 : An attacker cannot see when votes are swapped.

$$\mathrm{Voter}_1(0) \mid \mathrm{Voter}_2(1) \sim \mathrm{Voter}_1(1) \mid \mathrm{Voter}_2(0)$$

S. Kremer & M. Ryan

How to analyse security protocols ?

$\|$



$\overset{?}{\models}$

confidentiality
authenticity
anonymity
non-repudiation
...

Methodology

1. Proposing accurate models
   - symbolic models
   - cryptographic/computational models
2. Proving security
   - decision procedures
   - transfer results

How to analyse security protocols ?

How to prove e.g.

$$\forall A, \qquad A \mid \mathrm{Voter}_1(0) \mid \mathrm{Voter}_2(1) \sim A \mid \mathrm{Voter}_1(1) \mid \mathrm{Voter}_2(0)?$$

Task 2 : Automatic verification

How to analyse security protocols ?

How to prove e.g.

$$\forall A, \qquad A \mid \mathrm{Voter}_1(0) \mid \mathrm{Voter}_2(1) \sim A \mid \mathrm{Voter}_1(1) \mid \mathrm{Voter}_2(0)?$$

Task 2 : Automatic verification

- Unfortunately, security (e.g. confidentiality) is undecidable.
  $\longrightarrow$ No generic algorithm can work.
- Identification of decidable fragments
  - Analysis of a finite number of sessions
  - restriction on the class of protocols
- Semi-decision procedure : ProVerif

How does ProVerif work ?

Developed by Bruno Blanchet, ENS Paris, France.

- Implements a sound semi-decision procedure (that may not terminate).
- The applied pi-calculus is translated into first-order logic, more precisely into Horn clauses.
- Based on a resolution strategy well adapted to protocols.

Horn clauses for the intruder

Horn clauses perfectly reflect the attacker symbolic manipulations on terms.

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

Horn clauses for the protocol

Protocol WMF :		Horn clauses :		
$A \rightarrow S$:	$\{n_a, b, k\}_{k_a}$		$\Rightarrow$	$I(\{n_a, b, k\}_{k_a})$
$S \rightarrow B$:	$\{n_s, a, k\}_{k_b}$	$I(\{x, b, y\}_{k_a})$	$\Rightarrow$	$I(\{n_s(x, y), a, y\}_{k_b})$
$B \rightarrow A$:	$\{m_{ab}\}_k$	$I(\{x, a, y\}_{k_b})$	$\Rightarrow$	$I(\{m_{ab}\}_y)$

Horn clauses for the protocol

Protocol WMF :		Horn clauses :
$A \rightarrow S$: $\{n_a, b, k\}_{k_a}$		$\Rightarrow I(\{n_a, b, k\}_{k_a})$
$S \rightarrow B$: $\{n_s, a, k\}_{k_b}$		$I(\{x, b, y\}_{k_a}) \Rightarrow I(\{n_s(x, y), a, y\}_{k_b})$
$B \rightarrow A$: $\{m_{ab}\}_k$		$I(\{x, a, y\}_{k_b}) \Rightarrow I(\{m_{ab}\}_y)$

Secrecy property is a reachability (accessibility) property

$$\neg I(m_{ab})$$

Checking security reduces to checking satisfiability

There exists an attack iff the set of formulas corresponding to

Intruder manipulations + protocol + property

is NOT satisfiable.

How to decide satisfiability ?

→ Resolution techniques : Binary resolution

$$\frac{D_1 \wedge \cdots \wedge D_k \Rightarrow B \qquad A_1 \wedge \cdots \wedge A_n \Rightarrow C}{(D_1 \wedge \cdots \wedge D_k \wedge A_2 \wedge \cdots \wedge A_n \Rightarrow C)\theta} \; A_1\theta = B\theta$$

How to decide satisfiability ?

→ Resolution techniques : Binary resolution

$$\frac{D_1 \wedge \cdots \wedge D_k \Rightarrow B \qquad A_1 \wedge \cdots \wedge A_n \Rightarrow C}{(D_1 \wedge \cdots \wedge D_k \wedge A_2 \wedge \cdots \wedge A_n \Rightarrow C)\theta}\; A_1\theta = B\theta$$

→ It does not terminate.

Example :

$I(s) \qquad I(x), I(y) \Rightarrow I(\langle x, y\rangle)$

How to decide satisfiability ?

→ Resolution techniques : Binary resolution

$$\frac{D_1 \wedge \cdots \wedge D_k \Rightarrow B \qquad A_1 \wedge \cdots \wedge A_n \Rightarrow C}{(D_1 \wedge \cdots \wedge D_k \wedge A_2 \wedge \cdots \wedge A_n \Rightarrow C)\theta}\ A_1\theta = B\theta$$

→ It does not terminate.

Example :

$I(s) \qquad I(x), I(y) \Rightarrow I(\langle x, y \rangle)$

$I(y) \Rightarrow I(\langle s, y \rangle)$

How to decide satisfiability ?

→ Resolution techniques : Binary resolution

$$\frac{D_1 \wedge \cdots \wedge D_k \Rightarrow B \qquad A_1 \wedge \cdots \wedge A_n \Rightarrow C}{(D_1 \wedge \cdots \wedge D_k \wedge A_2 \wedge \cdots \wedge A_n \Rightarrow C)\theta} \; A_1\theta = B\theta$$

→ It does not terminate.

Example :

$I(s) \qquad I(x), I(y) \Rightarrow I(\langle x, y\rangle)$

$I(y) \Rightarrow I(\langle s, y\rangle)$

$I(\langle s, s\rangle)$

How to decide satisfiability ?

→ Resolution techniques : Binary resolution

$$\frac{D_1 \wedge \cdots \wedge D_k \Rightarrow B \qquad A_1 \wedge \cdots \wedge A_n \Rightarrow C}{(D_1 \wedge \cdots \wedge D_k \wedge A_2 \wedge \cdots \wedge A_n \Rightarrow C)\theta} \; A_1\theta = B\theta$$

→ It does not terminate.

Example :

$I(s) \qquad I(x), I(y) \Rightarrow I(\langle x, y\rangle)$

$I(y) \Rightarrow I(\langle s, y\rangle)$

$I(\langle s, s\rangle) \qquad I(\langle s, \langle s, s\rangle\rangle)$

How to decide satisfiability ?

$\rightarrow$ Resolution techniques : Binary resolution

$$\frac{D_1 \wedge \cdots \wedge D_k \Rightarrow B \qquad A_1 \wedge \cdots \wedge A_n \Rightarrow C}{(D_1 \wedge \cdots \wedge D_k \wedge A_2 \wedge \cdots \wedge A_n \Rightarrow C)\theta} \; A_1\theta = B\theta$$

$\rightarrow$ It does not terminate.

Example :

$I(s) \qquad I(x), I(y) \Rightarrow I(\langle x, y\rangle)$

$I(y) \Rightarrow I(\langle s, y\rangle)$

$I(\langle s, s\rangle) \qquad I(\langle s, \langle s, s\rangle\rangle) \qquad I(\langle s, \langle s, \langle s, s\rangle\rangle\rangle) \qquad I(\langle s, \langle s, \langle s, \langle s, s\rangle\rangle\rangle\rangle)$

$I(\langle s, \langle s, \langle s, \langle s, \langle s, s\rangle\rangle\rangle\rangle\rangle) \qquad \cdots$

Efficient and sound resolution strategy

Idea : Resolution is only applied on selected literals A_1, B that do not belong to a forbidden set S. Typically $S = \{I(x)\}$.

Theorem

Resolution based on selection, avoiding S, is complete w.r.t. satisfiability.

- If the fixed point does not contain the empty clause, then the corresponding protocol is secure.
- ProVerif may not terminate.

Efficient and sound resolution strategy

Idea : Resolution is only applied on selected literals A_1, B that do not belong to a forbidden set S. Typically $S = \{I(x)\}$.

Theorem

Resolution based on selection, avoiding S, is complete w.r.t. satisfiability.

- If the fixed point does not contain the empty clause, then the corresponding protocol is secure.
- ProVerif may not terminate.

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)
- Can handle various cryptographic primitives (various encryption, signatures, blind signatures, hash, *etc.*)

Security of Helios

$\rightarrow$ ProVerif cannot be applied (yet).

Privacy

$$\forall A, \quad A \mid \mathrm{Voter}_1(0) \mid \mathrm{Voter}_2(1) \sim A \mid \mathrm{Voter}_1(1) \mid \mathrm{Voter}_2(0)$$

Security of Helios

→ ProVerif cannot be applied (yet).

Privacy

$\forall A, \quad A \mid \mathsf{Voter}_1(0) \mid \mathsf{Voter}_2(1) \sim A \mid \mathsf{Voter}_1(1) \mid \mathsf{Voter}_2(0)$

- Helios is actually subject to replay attack, which breaks privacy !

Security of Helios

$\rightarrow$ ProVerif cannot be applied (yet).

Privacy

$\forall A, \quad A \mid \mathrm{Voter}_1(0) \mid \mathrm{Voter}_2(1) \sim A \mid \mathrm{Voter}_1(1) \mid \mathrm{Voter}_2(0)$

- Helios is actually subject to replay attack, which breaks privacy !
- The fixed version (weeding duplicated ballots) provably ensures privacy

Security of Helios

$\rightarrow$ ProVerif cannot be applied (yet).

Privacy

$$\forall A, \quad A \mid \mathrm{Voter}_1(0) \mid \mathrm{Voter}_2(1) \sim A \mid \mathrm{Voter}_1(1) \mid \mathrm{Voter}_2(0)$$

- Helios is actually subject to replay attack, which breaks privacy !
- The fixed version (weeding duplicated ballots) provably ensures privacy

Verifiability

- Individual verifiability : voter can check that her own ballot is included in the election's bulletin board.
- Universal verifiability : anyone can check that the election outcome corresponds to the ballots published on the bulletin board.

Helios provably satisfy both verifiability properties.

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 ?

→ Side channel attacks *cf* Seminar of Adi Shamir (May, 4th 2011)

→ Representing messages by a term algebra abstracts away many mathematical properties.

Setting for cryptographic/computational models

Messages : 01111001010110 (Bitstrings)

Protocol :

- Message exchange program
- Use cryptographic algorithms

cf Seminar of David Pointcheval (April, 27th 2011).

Setting for cryptographic/computational models

Messages : 01111001010110 (Bitstrings)

Protocol :

- Message exchange program
- Use cryptographic algorithms

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

cf Seminar of David Pointcheval (April, 27th 2011).

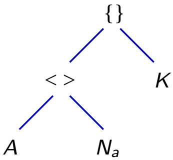
Formal and Cryptographic approaches

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

Link between the two approaches ?

Proving cryptographic security through symbolic models

Symbolic models

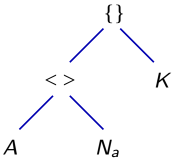


Computational models

0111010111010010
0101010001011101
1110010000110101

Proving cryptographic security through symbolic models

Symbolic models



Computational models

0111010111010010
0101010001011101
1110010000110101

Idea : soundness result

Show that security in symbolic models implies security in computational ones. [Abadi Rogaway 00]

Soundness of equivalences in the applied pi-calculus

Result : Assuming a strong encryption scheme (IND-CCA2 hypothesis)

$$P_1 \sim P_2 \quad \Rightarrow \quad [\![P_1]\!] \approx [\![P_2]\!]$$

Symbolic equivalence of processes P_1 and P_2	Indistinguishability of the implementation of P_1 and P_2

Soundness of equivalences in the applied pi-calculus

Result : Assuming a strong encryption scheme (IND-CCA2 hypothesis)

$$P_1 \sim P_2 \quad \Rightarrow \quad [\![P_1]\!] \approx [\![P_2]\!]$$

Symbolic equivalence of processes P_1 and P_2	Indistinguishability of the implementation of P_1 and P_2

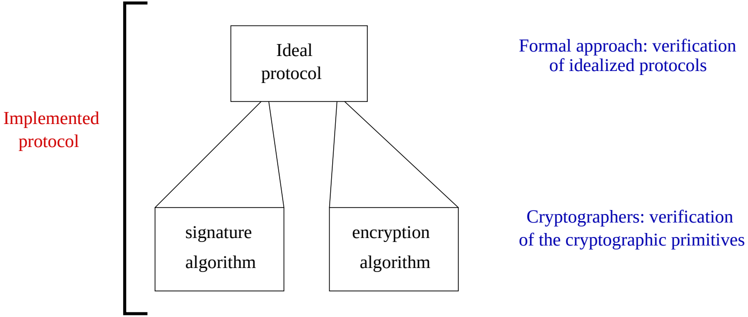
Key technique

Any attack trace from the concrete adversary is an attack against the symbolic protocol, or the adversary breaks encryption.

Consequence : Security in symbolic models directly implies security in cryptographic models, against arbitrary attackers.

Benefit : modularity

Cryptographic security guarantees can be obtained at the symbolic level



Conclusion

Formal methods form a powerful approach
for analyzing security protocols

- Use of existing techniques : term algebra, equational theories, clauses and resolution techniques, tree automata, etc.
  $\Rightarrow$ Many decision procedures
- Several successful automatic tools
  e.g. ProVerif, Avispa/Avantssar, Scyther, NRL Protocol Analyzer
  - Detect attacks (e.g. flaw in Gmail)
  - Prove security of standard protocols (e.g. IKE, JFK, Certified email, Helios, ...)
- Provides cryptographic guarantees under classical assumptions on the implementation of the primitives

The end

Special thanks to :

Hubert Comon-Lundh

Ben Smyth

Stéphanie Delaune

Bogdan Warinschi

Steve Kremer