Verification of security protocols: from confidentiality to privacy

Stéphanie Delaune

CNRS / IRISA, France

Monday, May 22nd, 2017

Cryptographic protocols everywhere !

$\longrightarrow$ they aim at securing communications over public networks

A variety of security properties

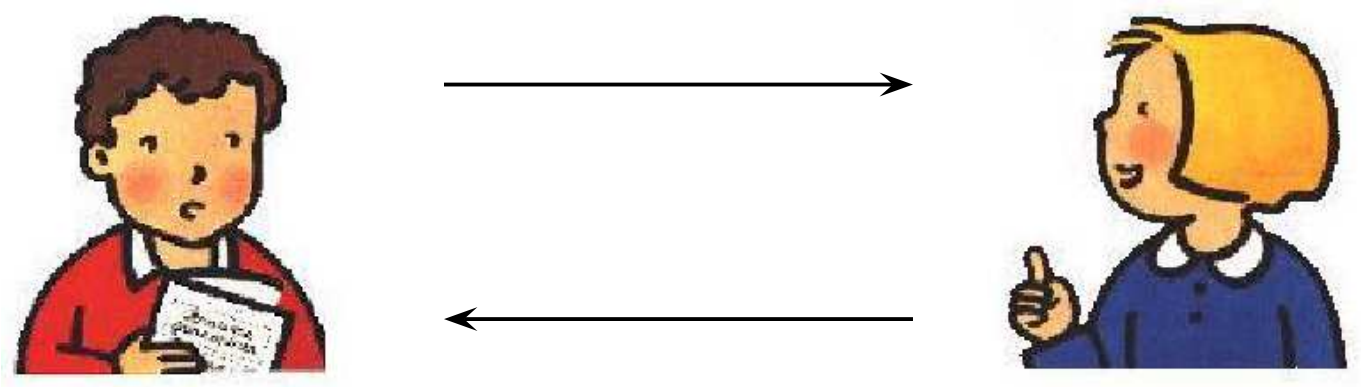
- **Secrecy**: May an intruder learn some secret message exchanged between two honest participants?
- Authentication: Is the agent Alice really talking to Bob?

A variety of security properties

- **Secrecy**: May an intruder learn some secret message exchanged between two honest participants?
- Authentication: Is the agent Alice really talking to Bob?
- Anonymity: Is an attacker able to learn something about the identity of the participants who are communicating?
- Non-repudiation: Alice sends a message to Bob. Alice cannot later deny having sent this message. Bob cannot deny having received the message.
- ...

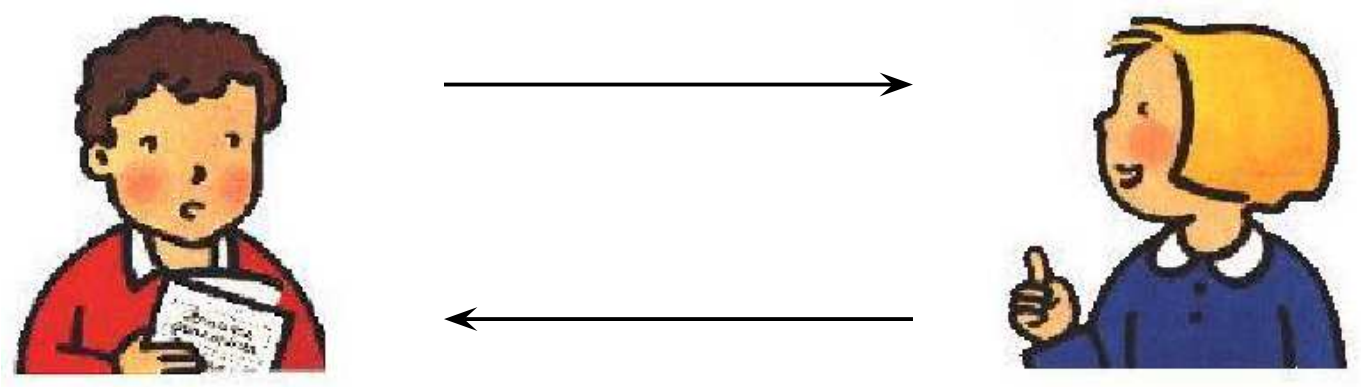
How does a cryptographic protocol work (or not)?

Protocol: small programs explaining how to exchange messages



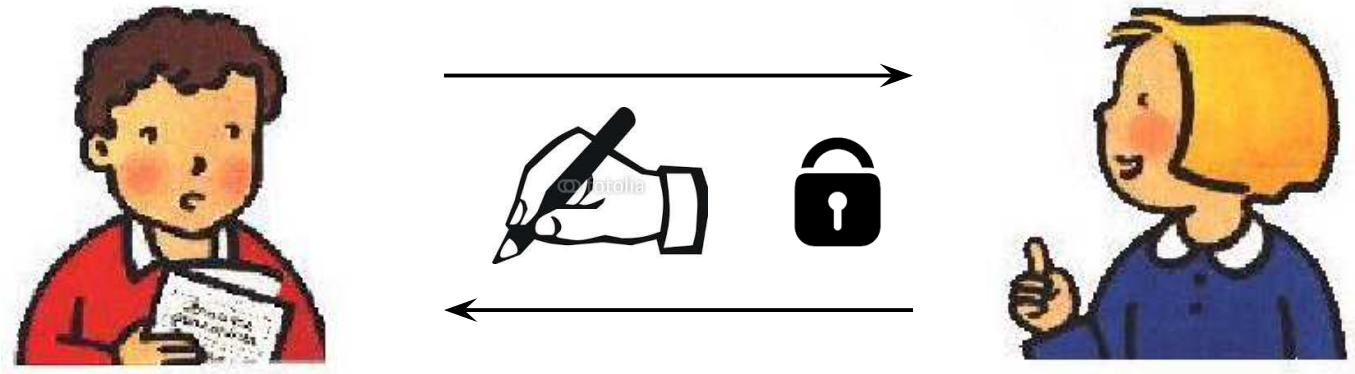
How does a cryptographic protocol work (or not)?

Protocol: small programs explaining how to exchange messages



How does a cryptographic protocol work (or not)?

Protocol: small programs explaining how to exchange messages

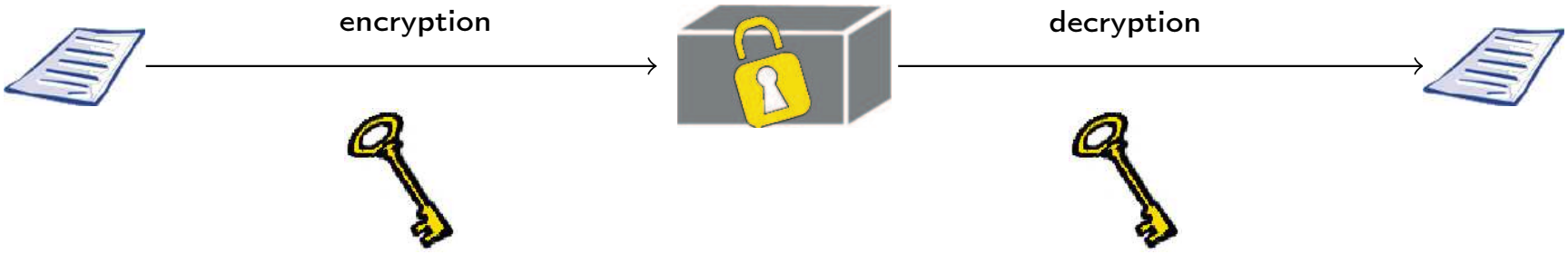


Cryptographic: make use of cryptographic primitives

Examples: symmetric encryption, asymmetric encryption, signature, hashes, ...

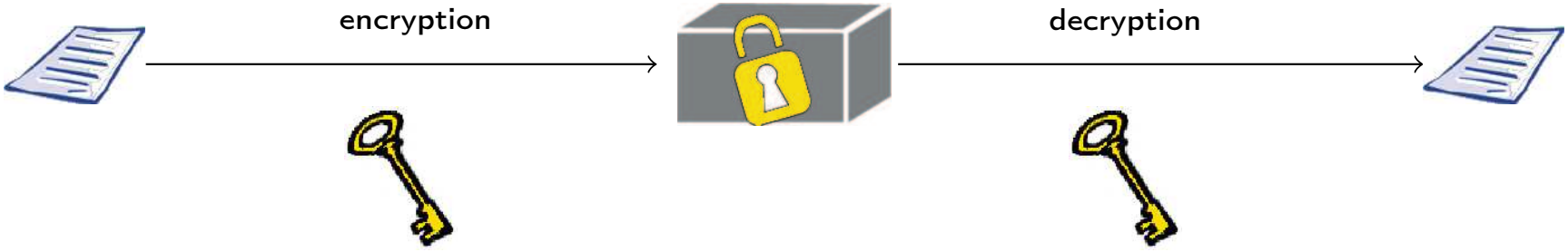
What is a symmetric encryption scheme?

Symmetric encryption

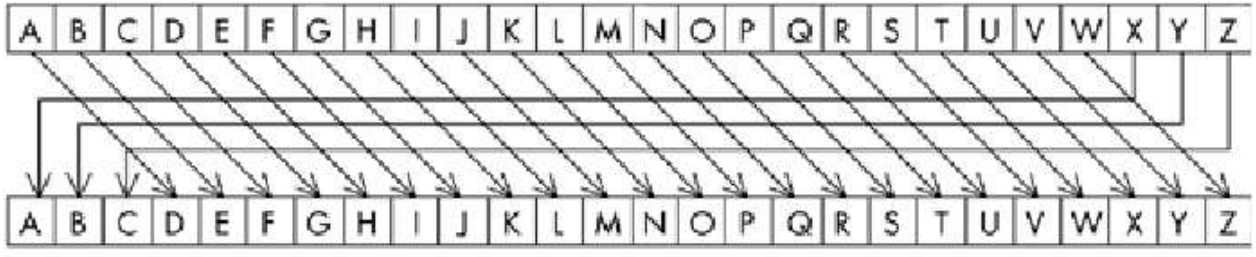


What is a symmetric encryption scheme?

Symmetric encryption



Example: This might be as simple as shifting each letter by a number of places in the alphabet (e.g. Caesar cipher)



Today: DES (1977), AES (2000)

A famous example

Enigma machine (1918-1945)

- electro-mechanical rotor cipher machines used by the German to encrypt during Wold War II
- permutations and substitutions

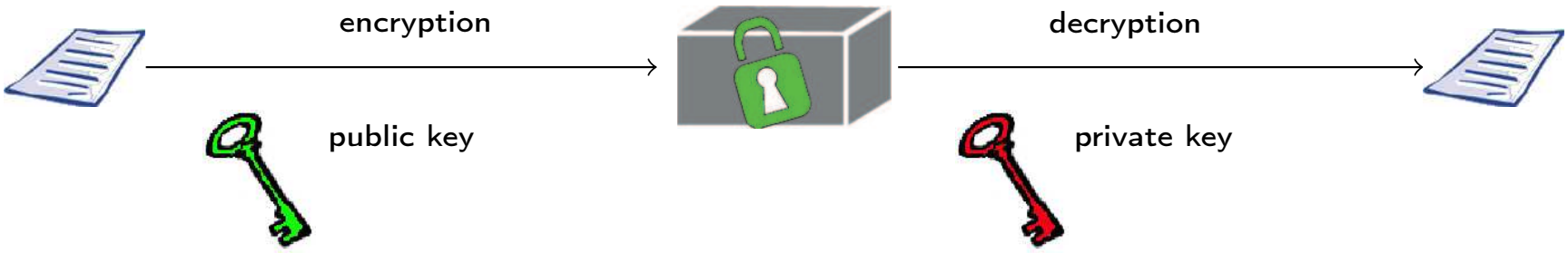
A bit of history

- 1918: invention of the Enigma machine
- 1940: Battle of the Atlantic during which Alan Turing's Bombe was used to test Enigma settings.

$\longrightarrow$ Everything about the breaking of the Enigma cipher systems remained secret until the mid-1970s.

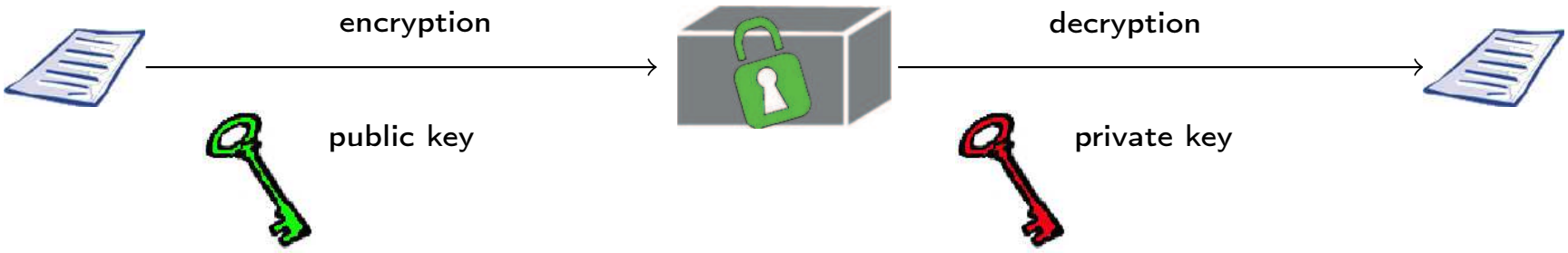
What is an asymmetric encryption scheme?

Asymmetric encryption



What is an asymmetric encryption scheme?

Asymmetric encryption



Examples:

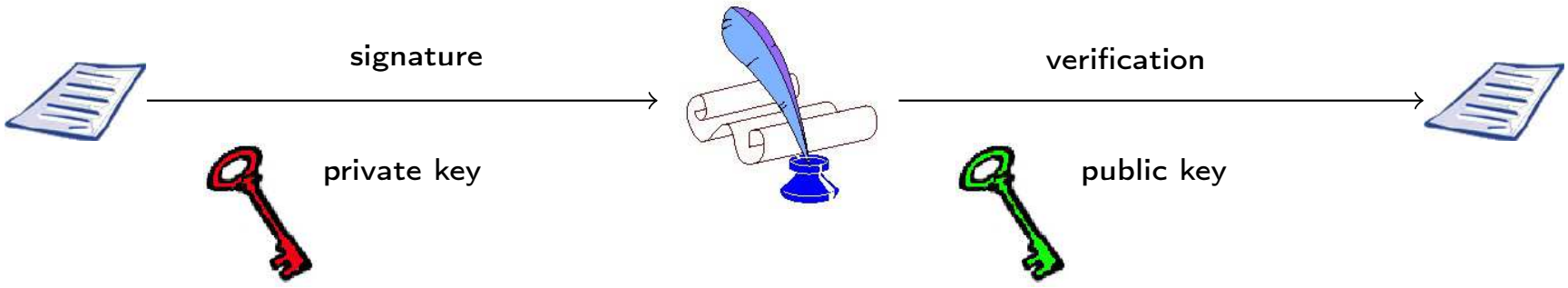
- 1976: first system published by W. Diffie, and M. Hellman,
- 1977: RSA system published by R. Rivest, A. Shamir, and L. Adleman.

$\longrightarrow$ their security relies on well-known mathematical problems (*e.g.* factorizing large numbers, computing discrete logarithms)

Today: those systems are still in use **Turing Award 2016**

What is a signature scheme?

Signature



Example:

The RSA cryptosystem (in fact, most public key cryptosystems) can be used as a signature scheme.

How cryptographic protocols can be attacked?

How cryptographic protocols can be attacked?

Logical attacks

- can be mounted even assuming perfect cryptography,
  $\hookrightarrow$ replay attack, man-in-the middle attack, ...
- subtle and hard to detect by "eyeballing" the protocol

How cryptographic protocols can be attacked?

Logical attacks

- can be mounted even assuming perfect cryptography,
  $\hookrightarrow$ replay attack, man-in-the middle attack, ...
- subtle and hard to detect by "eyeballing" the protocol

$\longrightarrow$ A traceability attack on the BAC protocol (2010)

Security

Defects in e-passports allow real-time tracking

This threat brought to you by RFID

The register - Jan. 2010

Example: Denning Sacco protocol (1981)

$$\xrightarrow{\text{aenc}(\text{sign}(k_{AB}, \text{priv}(A)), \text{pub}(B))}$$

Is the Denning Sacco protocol a good key exchange protocol?

Example: Denning Sacco protocol (1981)

$$\xrightarrow{\text{aenc}(\text{sign}(k_{AB}, \text{priv}(A)), \text{pub}(B))}$$

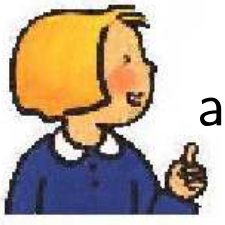
Is the Denning Sacco protocol a good key exchange protocol? No !

Example: Denning Sacco protocol (1981)

$$\xrightarrow{\text{aenc}(\text{sign}(k_{AB}, \text{priv}(A)), \text{pub}(B))}$$

Is the Denning Sacco protocol a good key exchange protocol? No !

Description of a possible attack:



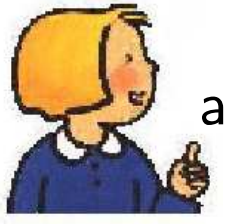
$$\xrightarrow{\text{aenc}(\text{sign}(k_{AC}, \text{priv}(A)), \text{pub}(C))}$$

Example: Denning Sacco protocol (1981)

$\text{aenc}(\text{sign}(k_{AB}, \text{priv}(A)), \text{pub}(B))$

Is the Denning Sacco protocol a good key exchange protocol? No !

Description of a possible attack:



$\text{aenc}(\text{sign}(k_{AC}, \text{priv}(A)), \text{pub}(C))$

$\text{sign}(k_{AC}, \text{priv}(A))$

k_{AC}

$\text{aenc}(\text{sign}(k_{AC}, \text{priv}(A)), \text{pub}(B))$

Exercise

We propose to fix the Denning-Sacco protocol as follows:

Version 1

$$A \to B \quad : \quad \mathsf{aenc}(\langle A, B, \mathsf{sign}(k, \mathsf{priv}(A))\rangle, \mathsf{pub}(B))$$

Version 2

$$A \to B \quad : \quad \mathsf{aenc}(\mathsf{sign}(\langle A, B, k\rangle, \mathsf{priv}(A))\rangle, \mathsf{pub}(B))$$

Which version would you prefer to use?

Exercise

We propose to fix the Denning-Sacco protocol as follows:

Version 1

$$A \to B \quad : \quad \mathsf{aenc}(\langle A, B, \mathsf{sign}(k, \mathsf{priv}(A))\rangle, \mathsf{pub}(B))$$

Version 2

$$A \to B \quad : \quad \mathsf{aenc}(\mathsf{sign}(\langle A, B, k\rangle, \mathsf{priv}(A))\rangle, \mathsf{pub}(B))$$

Which version would you prefer to use? Version 2

$\longrightarrow$ Version 1 is still vulnerable to the aforementioned attack.

What about protocols used in real life ?

Credit Card payment protocol

Serge Humpich case
“ Yescard “ (1997)

Credit Card payment protocol

Serge Humpich case
“ Yescard “ (1997)

Step 1: A logical flaw in the protocol allows one to copy a card and to use it without knowing the PIN code.

$\longrightarrow$ not a real problem, there is still a bank account to withdraw

Credit Card payment protocol

Serge Humpich case
“ Yescard “ (1997)

Step 1: A logical flaw in the protocol allows one to copy a card and to use it without knowing the PIN code.

$\longrightarrow$ not a real problem, there is still a bank account to withdraw

Step 2: breaking encryption via factorisation of the following (96 digits) number:
213598703592091008239502270499962879705109534182
641740644252416500858395774644508840500943086599 9

$\longrightarrow$ now, the number that is used is made of 232 digits

HTTPS connections

Lots of bugs and attacks, with fixes every month

HTTPS connections

Lots of bugs and attacks, with fixes every month

FREAK attack discovered by Baraghavan et al (Feb. 2015)

1. a logical flaw that allows a man in the middle attacker to downgrade connections from 'strong' RSA to 'export-grade' RSA;
2. breaking encryption via factorisation of such a key can be easily done.

HTTPS connections

Lots of bugs and attacks, with fixes every month

FREAK attack discovered by Baraghavan et al (Feb. 2015)

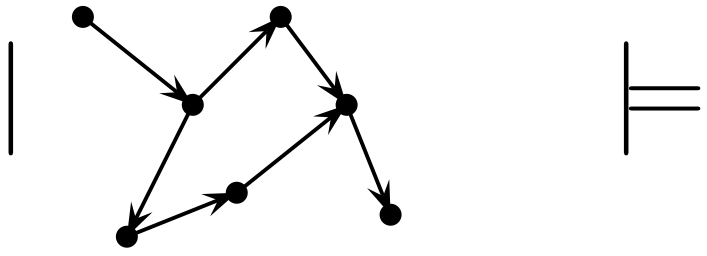
1. a logical flaw that allows a man in the middle attacker to downgrade connections from 'strong' RSA to 'export-grade' RSA;
2. breaking encryption via factorisation of such a key can be easily done.

$\longrightarrow$ 'export-grade' were introduced under the pressure of US governments agencies to ensure that they would be able to decrypt all foreign encrypted communication.

This talk: formal methods for protocol verification

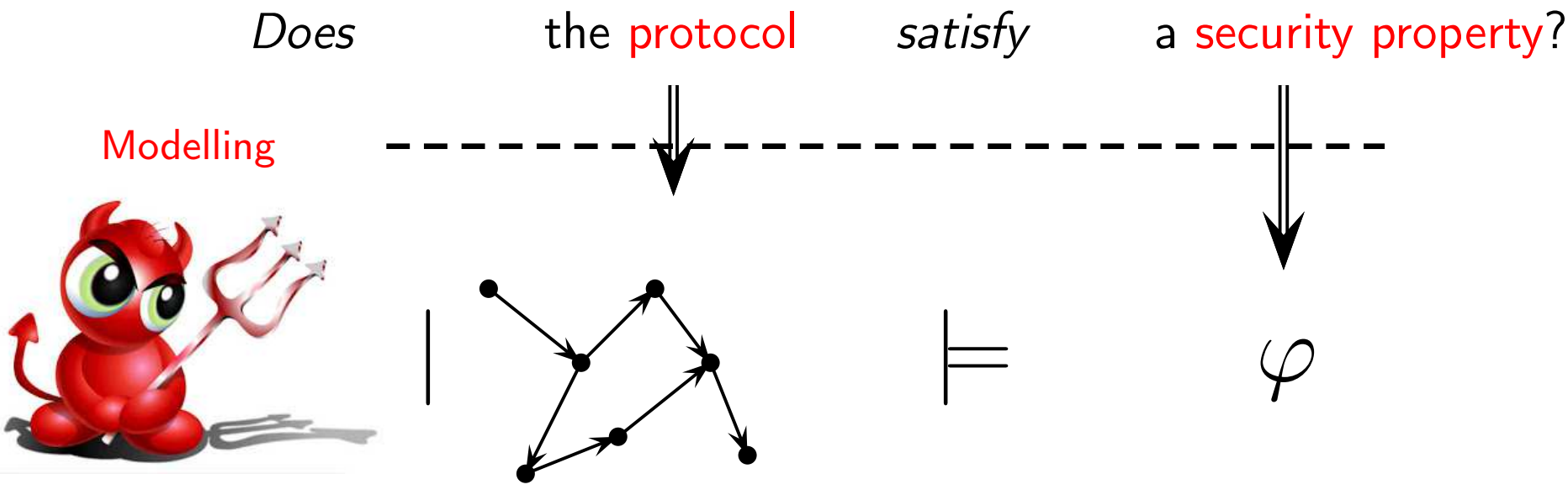
Does the protocol *satisfy* a security property?

Modelling



$\models$ φ

This talk: formal methods for protocol verification



Outline of the this talk

1. Modelling protocols, security properties, and the attacker
2. Designing verification algorithms

Part I

Modelling protocols, security properties and the attacker

Two major families of models ...

... with some advantages and some drawbacks.

Computational model

- + messages are bitstring, a general and powerful adversary
- − manual proofs, tedious and error-prone

Symbolic model

- − abstract model, e.g. messages are terms
- + automatic proofs

Two major families of models ...

... with some advantages and some drawbacks.

Computational model

- + messages are bitstring, a general and powerful adversary
- − manual proofs, tedious and error-prone

Symbolic model

- − abstract model, e.g. messages are terms
- + automatic proofs

Some results allowed to make a link between these two very different models.

$\longrightarrow$ Abadi & Rogaway 2000

Protocols as processes

Applied pi calculus [Abadi & Fournet, 01]

basic programming language with constructs for concurrency and communication

$\longrightarrow$ based on the π-calculus [Milner *et al.*, 92] ...

P, Q	$:=$	0	null process
		$\mathsf{in}(c, x).P$	input
		$\mathsf{out}(c, u).P$	output
		$\mathsf{if}\ u = v\ \mathsf{then}\ P\ \mathsf{else}\ Q$	conditional
		$P \mid Q$	parallel composition
		$!P$	replication
		$\mathsf{new}\ n.P$	fresh name generation

Protocols as processes

Applied pi calculus [Abadi & Fournet, 01]

basic programming language with constructs for concurrency and communication

$\longrightarrow$ based on the π-calculus [Milner *et al.*, 92] ...

P, Q	$:=$	0	null process
		$\mathsf{in}(c, x).P$	input
		$\mathsf{out}(c, u).P$	output
		$\mathsf{if}\ u = v\ \mathsf{then}\ P\ \mathsf{else}\ Q$	conditional
		$P \mid Q$	parallel composition
		$!P$	replication
		$\mathsf{new}\ n.P$	fresh name generation

... but messages that are exchanged are not necessarily atomic !

Messages as terms

Terms are built over a set of names $\mathcal{N}$, and a signature $\mathcal{F}$.

$$
\begin{array}{lcll}
\mathsf{t} & ::= & n & \text{name } n \\
 & | & f(t_1, \ldots, t_k) & \text{application of symbol } f \in \mathcal{F}
\end{array}
$$

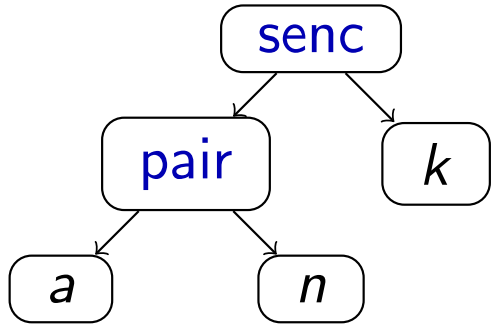
Messages as terms

Terms are built over a set of names $\mathcal{N}$, and a signature $\mathcal{F}$.

$$\begin{array}{lcll} \mathsf{t} & ::= & n & \text{name } n \\ & | & f(t_1, \ldots, t_k) & \text{application of symbol } f \in \mathcal{F} \end{array}$$

Example: representation of $\{a, n\}_k$

- Names: n, k, a
- constructors: senc, pair,



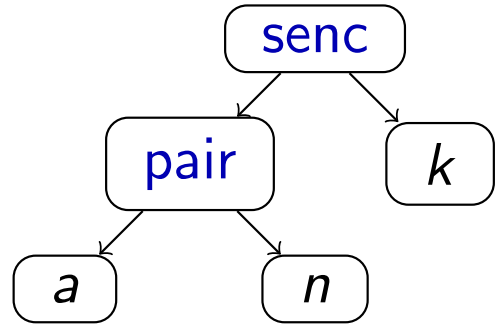
Messages as terms

Terms are built over a set of names $\mathcal{N}$, and a signature $\mathcal{F}$.

$$\begin{array}{lcll} \mathrm{t} & ::= & n & \text{name } n \\ & | & f(t_1, \ldots, t_k) & \text{application of symbol } f \in \mathcal{F} \end{array}$$

Example: representation of $\{a, n\}_k$

- Names: n, k, a
- constructors: senc, pair,
- destructors: sdec, proj_1, proj_2.



The term algebra is equipped with an equational theory E.

$$\mathrm{sdec}(\mathrm{senc}(x, y), y) = x \qquad \begin{array}{rcl} \mathrm{proj}_1(\mathrm{pair}(x, y)) & = & x \\ \mathrm{proj}_2(\mathrm{pair}(x, y)) & = & y \end{array}$$

Example: $\mathrm{sdec}(\mathrm{senc}(s, k), k) =_{\mathsf{E}} s$.

Semantics

Semantics $\rightarrow$:

Comm	$\mathsf{out}(c, u).P \mid \mathsf{in}(c, x).Q \rightarrow P \mid Q\{u/x\}$
Then	$\mathsf{if}\ u = v\ \mathsf{then}\ P\ \mathsf{else}\ Q \rightarrow P\ \mathsf{when}\ u =_{\mathsf{E}} v$
Else	$\mathsf{if}\ u = v\ \mathsf{then}\ P\ \mathsf{else}\ Q \rightarrow Q \quad \mathsf{when}\ u \neq_{\mathsf{E}} v$

Semantics

Semantics $\rightarrow$:

Comm	$\mathsf{out}(c, u).P \mid \mathsf{in}(c, x).Q \rightarrow P \mid Q\{u/x\}$
Then	$\mathsf{if}\ u = v\ \mathsf{then}\ P\ \mathsf{else}\ Q \rightarrow P$ when $u =_{\mathsf{E}} v$
Else	$\mathsf{if}\ u = v\ \mathsf{then}\ P\ \mathsf{else}\ Q \rightarrow Q$ when $u \neq_{\mathsf{E}} v$

closed by

- structural equivalence ($\equiv$):

$$P \mid Q \equiv Q \mid P, \qquad P \mid 0 \equiv P, \qquad \ldots$$

- application of evaluation contexts:

$$\frac{P \rightarrow P'}{\mathsf{new}n.\, P \rightarrow \mathsf{new}n.\, P'} \qquad \frac{P \rightarrow P'}{P \mid Q \rightarrow P' \mid Q}$$

Going back to the Denning Sacco protocol (1/3)

$$\begin{array}{lcl} A \rightarrow B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \rightarrow A & : & \mathsf{senc}(s, k) \end{array}$$

What symbols and equations do we need to model this protocol?

Going back to the Denning Sacco protocol (1/3)

$$\begin{array}{lcl} A \rightarrow B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \rightarrow A & : & \mathsf{senc}(s, k) \end{array}$$

What symbols and equations do we need to model this protocol?

1. symmetric encryption: senc and sdec

$$\mathsf{sdec}(\mathsf{senc}(x, y), y) = x$$

Going back to the Denning Sacco protocol (1/3)

$$\begin{array}{lcl} A \rightarrow B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \rightarrow A & : & \mathsf{senc}(s, k) \end{array}$$

What symbols and equations do we need to model this protocol?

1. symmetric encryption: senc and sdec

$$\mathsf{sdec}(\mathsf{senc}(x, y), y) = x$$

2. asymmetric encryption: aenc, adec, and pk

$$\mathsf{adec}(\mathsf{aenc}(x, \mathsf{pk}(y)), y) = x$$

Going back to the Denning Sacco protocol (1/3)

$$\begin{array}{lcl} A \rightarrow B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \rightarrow A & : & \mathsf{senc}(s, k) \end{array}$$

What symbols and equations do we need to model this protocol?

1. symmetric encryption: senc and sdec

$$\mathsf{sdec}(\mathsf{senc}(x, y), y) = x$$

2. asymmetric encryption: aenc, adec, and pk

$$\mathsf{adec}(\mathsf{aenc}(x, \mathsf{pk}(y)), y) = x$$

3. signature: ok, sign, check, getmsg, and pk

$$\mathsf{check}(\mathsf{sign}(x, y), \mathsf{pk}(y)) = \mathsf{ok} \text{ and } \mathsf{getmsg}(\mathsf{sign}(x, y)) = x$$

Going back to the Denning Sacco protocol (1/3)

$$\begin{array}{lcl} A \rightarrow B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \rightarrow A & : & \mathsf{senc}(s, k) \end{array}$$

What symbols and equations do we need to model this protocol?

1. symmetric encryption: senc and sdec

$$\mathsf{sdec}(\mathsf{senc}(x, y), y) = x$$

2. asymmetric encryption: aenc, adec, and pk

$$\mathsf{adec}(\mathsf{aenc}(x, \mathsf{pk}(y)), y) = x$$

3. signature: ok, sign, check, getmsg, and pk

$$\mathsf{check}(\mathsf{sign}(x, y), \mathsf{pk}(y)) = \mathsf{ok} \text{ and } \mathsf{getmsg}(\mathsf{sign}(x, y)) = x$$

The two terms involved in a normal execution are:

$$\mathsf{aenc}(\mathsf{sign}(k, ska), \mathsf{pk}(skb)), \quad \text{and} \quad \mathsf{senc}(s, k)$$

Going back to the Denning Sacco protocol (2/3)

$$
\begin{array}{lcl}
A \rightarrow B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\
B \rightarrow A & : & \mathsf{senc}(s, k)
\end{array}
$$

Going back to the Denning Sacco protocol (2/3)

$$\begin{array}{lcl} A \to B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \to A & : & \mathsf{senc}(s, k) \end{array}$$

Alice and Bob as processes:

$$\begin{array}{lcl} P_A(sk_a, pk_b) & = & \mathsf{new}\ k. \\ & & \mathsf{out}(c, \mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b)). \\ & & \mathsf{in}(c, x_a).\ \ldots \end{array}$$

Going back to the Denning Sacco protocol (2/3)

$$\begin{array}{lcl} A \to B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \to A & : & \mathsf{senc}(s, k) \end{array}$$

Alice and Bob as processes:

$$\begin{array}{lcl} P_A(sk_a, pk_b) & = & \mathsf{new}\ k. \\ & & \mathsf{out}(c, \mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b)). \\ & & \mathsf{in}(c, x_a).\ \ldots \end{array}$$

$$\begin{array}{lcl} P_B(sk_b, pk_a) & = & \mathsf{in}(c, x_b). \\ & & \ \ \mathsf{if}\ \mathsf{check}(\mathsf{adec}(x_b, sk_b), pk_a) = \mathsf{ok}\ \mathsf{then} \\ & & \qquad \mathsf{new}\ s. \\ & & \qquad \mathsf{out}(c, \mathsf{senc}(s, \mathsf{getmsg}(\mathsf{adec}(x_b, sk_b)))) \end{array}$$

Going back to the Denning Sacco protocol (3/3)

$P_A(sk_a, pk_b) =$
new k.
$\mathsf{out}(c, \mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b))$.
$\mathsf{in}(c, x_a)$. ...

$P_B(sk_b, pk_a) =$
$\mathsf{in}(c, x_b)$.
if $\mathsf{check}(\mathsf{adec}(x_b, sk_b), pk_a) = \mathsf{ok}$ then
new s.
$\mathsf{out}(c, \mathsf{senc}(s, \mathsf{getmsg}(\mathsf{adec}(x_b, sk_b))))$

Going back to the Denning Sacco protocol (3/3)

$P_A(sk_a, pk_b) =$
new k.
$\mathsf{out}(c, \mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b))$.
$\mathsf{in}(c, x_a)$. ...

$P_B(sk_b, pk_a) =$
$\mathsf{in}(c, x_b)$.
if $\mathsf{check}(\mathsf{adec}(x_b, sk_b), pk_a) = \mathsf{ok}$ then
new s.
$\mathsf{out}(c, \mathsf{senc}(s, \mathsf{getmsg}(\mathsf{adec}(x_b, sk_b))))$

We consider the following scenario:

$$P_{\mathsf{DS}} = \text{new } sk_a, sk_b.\big(P_A(sk_a, \mathsf{pk}(sk_b)) \mid P_B(sk_b, \mathsf{pk}(sk_a))\big)$$
$$\rightarrow \text{new } sk_a, sk_b, k.\big(\ \mathsf{in}(c, x_a).\ \ldots$$
$$\mid\ \text{if } \mathsf{check}(\mathsf{adec}(\mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b), sk_b), pk_a) = \mathsf{ok} \text{ then}$$
$$\text{new } s.\mathsf{out}(c, \mathsf{senc}(s, \mathsf{getmsg}(\mathsf{adec}(\mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b), sk_b)))))\big)$$

Going back to the Denning Sacco protocol (3/3)

$P_A(sk_a, pk_b) =$
new k.
$\mathsf{out}(c, \mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b))$.
$\mathsf{in}(c, x_a)$. ...

$P_B(sk_b, pk_a) =$
$\mathsf{in}(c, x_b)$.
if $\mathsf{check}(\mathsf{adec}(x_b, sk_b), pk_a) = \mathsf{ok}$ then
new s.
$\mathsf{out}(c, \mathsf{senc}(s, \mathsf{getmsg}(\mathsf{adec}(x_b, sk_b))))$

We consider the following scenario:

$P_{\mathsf{DS}} = \text{new } sk_a, sk_b.(P_A(sk_a, \mathsf{pk}(sk_b)) \mid P_B(sk_b, \mathsf{pk}(sk_a)))$

$\rightarrow \text{new } sk_a, sk_b, k.(\ \mathsf{in}(c, x_a).\ \ldots$
$\mid$ if $\mathsf{check}(\mathsf{adec}(\mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b), sk_b), pk_a) = \mathsf{ok}$ then
new s.$\mathsf{out}(c, \mathsf{senc}(s, \mathsf{getmsg}(\mathsf{adec}(\mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b), sk_b)))))$

$\rightarrow \text{new } sk_a, sk_b, k.(\ \mathsf{in}(c, x_a).\ \ldots$
new s.$\mathsf{out}(c, \mathsf{senc}(s, \underline{\mathsf{getmsg}(\mathsf{adec}(\mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b), sk_b))}))$

Going back to the Denning Sacco protocol (3/3)

$P_A(sk_a, pk_b) =$
new k.
$\mathsf{out}(c, \mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b))$.
$\mathsf{in}(c, x_a)$. ...

$P_B(sk_b, pk_a) =$
$\mathsf{in}(c, x_b)$.
if $\mathsf{check}(\mathsf{adec}(x_b, sk_b), pk_a) = \mathsf{ok}$ then
new s.
$\mathsf{out}(c, \mathsf{senc}(s, \mathsf{getmsg}(\mathsf{adec}(x_b, sk_b))))$

We consider the following scenario:

$P_{\mathsf{DS}} = \text{new } sk_a, sk_b.(P_A(sk_a, \mathsf{pk}(sk_b)) \mid P_B(sk_b, \mathsf{pk}(sk_a))$

$\rightarrow \text{new } sk_a, sk_b, k.(\ \mathsf{in}(c, x_a).\ \ldots$
$\mid$ if $\mathsf{check}(\mathsf{adec}(\mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b), sk_b), pk_a) = \mathsf{ok}$ then
new s.$\mathsf{out}(c, \mathsf{senc}(s, \mathsf{getmsg}(\mathsf{adec}(\mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b), sk_b)))))$

$\rightarrow \text{new } sk_a, sk_b, k.(\ \mathsf{in}(c, x_a).\ \ldots$
new s.$\mathsf{out}(c, \mathsf{senc}(s, \underline{\mathsf{getmsg}(\mathsf{adec}(\mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b), sk_b))}))))$

$\longrightarrow$ this derivation represents a normal execution between two honest participants

Security properties - confidentiality

Confidentiality for process P w.r.t. secret s

For all processes A such that $A \mid P \rightarrow^* Q$, we have that Q is not of the form $C[\mathrm{out}(c, s).Q']$ with c public.

Security properties - confidentiality

Confidentiality for process P w.r.t. secret s

For all processes A such that $A \mid P \rightarrow^* Q$, we have that Q is not of the form $C[\mathrm{out}(c, s).Q']$ with c public.

Some difficulties:

- we have to consider all the possible executions in presence of an arbitrary adversary (modelled as a process)
- we have to consider realistic initial configurations
  - an unbounded number of agents,
  - replications to model an unbounded number of sessions,
  - reveal public keys and private keys to model dishonest agents,
  - honest agents may initiate a session with a dishonest agent, ...

Going back to the Denning Sacco protocol

$$\begin{array}{lcl} A \to B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \to A & : & \mathsf{senc}(s, k) \end{array}$$

The aforementioned attack

1. $A \to C$: $\mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(C))$

2. $C(A) \to B$: $\mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B))$
3. $B \to A$: $\mathsf{senc}(s, k)$

The "minimal" initial configuration to retrieve the attack is:

$$\mathsf{new}\ sk_a, sk_b.\big(P_A(sk_a, \mathsf{pk}(sk_c)) \mid P_B(sk_b, \mathsf{pk}(sk_a) \mid \mathsf{out}(c, \mathsf{pk}(skb)))\big)$$

Going back to the Denning Sacco protocol

$$\begin{array}{lcl} A \to B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \to A & : & \mathsf{senc}(s, k) \end{array}$$

The aforementioned attack

$$\begin{array}{l} 1.\ A \to C : \ \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(C)) \\ \qquad\qquad 2.\ C(A) \to B : \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ \qquad\qquad 3.\ \quad B \to A : \mathsf{senc}(s, k) \end{array}$$

The "minimal" initial configuration to retrieve the attack is:

$$\mathsf{new}\ sk_a, sk_b.\big(P_A(sk_a, \mathsf{pk}(sk_c)) \mid P_B(sk_b, \mathsf{pk}(sk_a) \mid \mathsf{out}(c, \mathsf{pk}(skb)))\big)$$

Exercise: Exhibit the process A (the behaviour of the attacker) that witnesses the aforementioned attack, i.e. such that:

$$A \mid P_{\mathsf{DS}} \ \to^* \ C[\mathsf{out}(c, s).Q']$$

Security properties - authentication (1/2)

Authentication is usually expressed as a correspondence property.

Intuitively, if B finishes a session, thinking he has talked to A (with session nonces $N_1, \ldots$) then A has also finished a session, thinking she has talked to B (with session nonces $N_1, \ldots$).

Security properties - authentication (1/2)

Authentication is usually expressed as a correspondence property.

Intuitively, if B finishes a session, thinking he has talked to A (with session nonces $N_1, \ldots$) then A has also finished a session, thinking she has talked to B (with session nonces $N_1, \ldots$).

More formally, we enrich the syntax/semantics of our process algebra:

$$
\begin{aligned}
&P, Q, R := \\
&\quad \mathsf{in}(c, x).P \\
&\quad \mathsf{out}(c, N).P \\
&\quad \ldots \\
&\quad \mathsf{event}(p(u_1, \ldots, u_n)).P
\end{aligned}
$$

EVENT $\quad \mathsf{event}(p(u_1, \ldots, u_n)).P \xrightarrow{\mathsf{event}(p(u_1,\ldots,u_n))} P$

Security properties - authentication (2/2)

Given a process P, and a security formula F of the form

$$\text{for all } \vec{x} \qquad \text{event}(p(u_1, \ldots, u_n)) \Rightarrow \text{event}(q(v_1, \ldots, v_k))$$

Security properties - authentication (2/2)

Given a process P, and a security formula F of the form

$$\text{for all } \vec{x} \qquad \text{event}(p(u_1, \ldots, u_n)) \Rightarrow \text{event}(q(v_1, \ldots, v_k))$$

we say that P satisfies the formula F if for all processes A such that

$$A \mid P \xrightarrow{\alpha_1 \ldots \alpha_n} Q$$

we have that whenever $\alpha_i = p(u_1, \ldots, u_n)\theta$ for some θ

then there exists $j < i$ such that

$$\alpha_j = q(v_1, \ldots, v_k)\theta$$

Intuitively: whenever $\text{event}(p(u_1, \ldots, u_n))$ *occurs, a corresponding* $\text{event}(q(v_1, \ldots, v_k))$ *has occurred before.*

Part II

Designing verification algorithms

confidentiality, authentication

State of the art in a nutshell

for analysing confidentiality/authentication properties

Unbounded number of sessions

- undecidable in general [Even & Goldreich, 83; Durgin *et al*, 99]
- decidable for restricted classes [Lowe, 99] [Rammanujam & Suresh, 03]

$\longrightarrow$ existing verification tools: ProVerif, Tamarin, Maude-NPA, ...

State of the art in a nutshell

for analysing confidentiality/authentication properties

Unbounded number of sessions

- undecidable in general [Even & Goldreich, 83; Durgin *et al*, 99]
- decidable for restricted classes [Lowe, 99] [Rammanujam & Suresh, 03]

$\longrightarrow$ existing verification tools: ProVerif, Tamarin, Maude-NPA, ...

Bounded number of sessions

- a decidability result (NP-complete)
  [Rusinowitch & Turuani, 01; Millen & Shmatikov, 01]
- result extended to deal with various cryptographic primitives.

$\longrightarrow$ automatic tools, e.g. AVISPA platform [Armando *et al.*, 05]

ProVerif [Blanchet, 01]

ProVerif is a verifier for cryptographic protocols that may prove that a protocol is secure or exhibit attacks.

`http://proverif.inria.fr`

Advantages

- fully automatic, and quite efficient
- a rich process algebra: replication, else branches, ...
- handles many cryptographic primitives
- various security properties: secrecy, correspondences, equivalences

ProVerif [Blanchet, 01]

ProVerif is a verifier for cryptographic protocols that may prove that a protocol is secure or exhibit attacks.

`http://proverif.inria.fr`

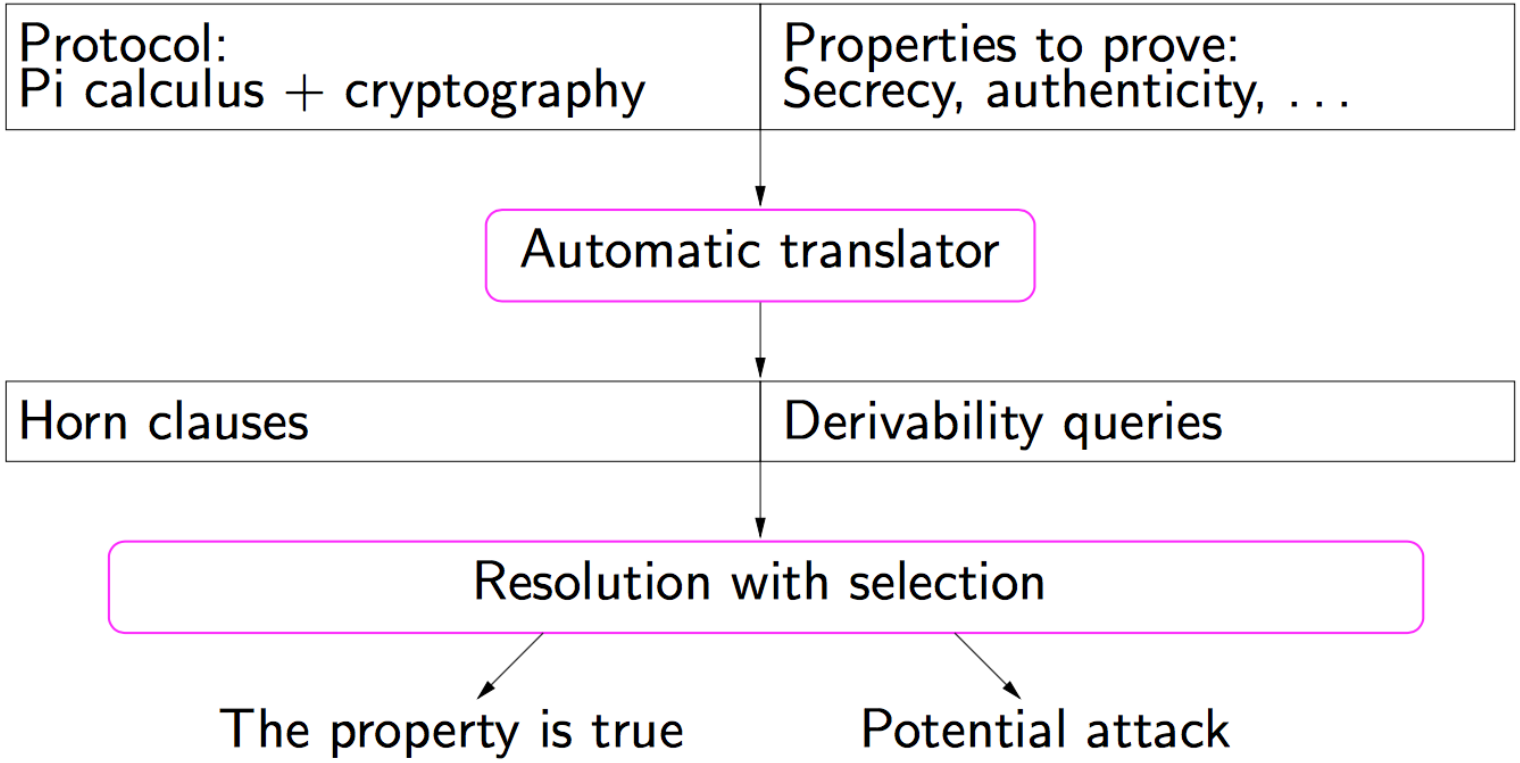
Advantages

- fully automatic, and quite efficient
- a rich process algebra: replication, else branches, ...
- handles many cryptographic primitives
- various security properties: secrecy, correspondences, equivalences

No miracle

- the tool can say "can not be proved";
- termination is not guaranteed

How does ProVerif work?



▸ Skip details

Some vocabulary

First order logic

Atoms	$P(t_1, \ldots, t_n)$ where t_i are terms, P is a predicate
Literals	$P(t_1, \ldots, t_n)$ or $\neg P(t_1, \ldots, t_n)$
closed under	$\vee, \wedge, \neg, \exists, \forall$

Clauses: Only universal quantifiers

Horn Clauses: at most one positive literal (where A_i, B are atoms.)

$$\forall \tilde{x}.\ A_1, \ldots, A_n \Rightarrow B$$

Modelling the attacker using Horn clauses

Public key encryption

$$\begin{aligned} \mathsf{att}(x) &\Rightarrow \mathsf{att}(\mathsf{pk}(x)) \\ \mathsf{att}(x),\ \mathsf{att}(\mathsf{pk}(y)) &\Rightarrow \mathsf{att}(\mathsf{aenc}(x, \mathsf{pk}(y))) \\ \mathsf{att}((\mathsf{aenc}(x, \mathsf{pk}(y))),\ \mathsf{att}(y) &\Rightarrow \mathsf{att}(x) \end{aligned}$$

Signature

$$\begin{aligned} \mathsf{att}(x),\ \mathsf{att}(y) &\Rightarrow \mathsf{att}(\mathsf{sign}(x, y)) \\ \mathsf{att}(\mathsf{sign}(x, y)) &\Rightarrow \mathsf{att}(x) \end{aligned}$$

Symmetric encryption

$$\begin{aligned} \mathsf{att}(x),\ \mathsf{att}(y) &\Rightarrow \mathsf{att}(\mathsf{senc}(x, y)) \\ \mathsf{att}((\mathsf{senc}(x, y)),\ \mathsf{att}(y) &\Rightarrow \mathsf{att}(x) \end{aligned}$$

Initial knowledge

$$\Rightarrow \mathsf{att}(\mathsf{pk}(sk_A)) \qquad \Rightarrow \mathsf{att}(sk_I) \qquad \Rightarrow \mathsf{att}(\mathsf{pk}(sk_B))$$

Modelling the protococol using Horn clauses

Denning-Sacco protocol ...

$$\begin{array}{lcl} A \to B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \to A & : & \mathsf{senc}(s, k) \end{array}$$

... using Horn clauses

- A talks with any principal represented by its public key $\mathsf{pk}(x)$.

$$\mathsf{att}(\mathsf{pk}(x)) \Rightarrow \mathsf{att}(\mathsf{aenc}(\mathsf{sign}(k, sk_A), \mathsf{pk}(x)))$$

- When B receives a message of the expected form, he replies accordingly

$$\mathsf{att}(\mathsf{aenc}(\mathsf{sign}(y, sk_A), \mathsf{pk}(sk_B))) \Rightarrow \mathsf{att}(s, y)$$

Modelling the protococol using Horn clauses

Denning-Sacco protocol ...

$$\begin{array}{lcl} A \to B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \to A & : & \mathsf{senc}(s, k) \end{array}$$

... using Horn clauses

- A talks with any principal represented by its public key $\mathsf{pk}(x)$.

$$\mathsf{att}(\mathsf{pk}(x)) \Rightarrow \mathsf{att}(\mathsf{aenc}(\mathsf{sign}(k[x], sk_A), \mathsf{pk}(x)))$$

- When B receives a message of the expected form, he replies accordingly

$$\mathsf{att}(\mathsf{aenc}(\mathsf{sign}(y, sk_A), \mathsf{pk}(sk_B))) \Rightarrow \mathsf{att}(s, y)$$

$\longrightarrow$ names are **parametrized** to partially modelled their freshness

Modelling the security property using Horn clauses

We consider secrecy as a reachability (accessibility) property.

Is $\mathcal{C}_{att} + \mathcal{C}_{prot} + \neg\text{att}(s)$ satisfiable or not?

Modelling the security property using Horn clauses

We consider secrecy as a reachability (accessibility) property.

Is $\mathcal{C}_{att} + \mathcal{C}_{prot} + \neg\text{att}(s)$ satisfiable or not?

Denning Sacco protocol

$\text{att}(sk_I)$	initial knowledge

Modelling the security property using Horn clauses

We consider secrecy as a reachability (accessibility) property.

Is $\mathcal{C}_{att} + \mathcal{C}_{prot} + \neg\text{att}(s)$ satisfiable or not?

Denning Sacco protocol

$\text{att}(sk_I)$	initial knowledge
$\text{att}(\text{pk}(sk_I))$	using attacker rules

Modelling the security property using Horn clauses

We consider secrecy as a reachability (accessibility) property.

Is $\mathcal{C}_{att} + \mathcal{C}_{prot} + \neg\text{att}(s)$ satisfiable or not?

Denning Sacco protocol

$\text{att}(sk_I)$	initial knowledge
$\text{att}(\text{pk}(sk_I))$	using attacker rules
$\text{att}(\text{aenc}(\text{sign}(k[sk_I], sk_A), \text{pk}(sk_I)))$	using protocol (rule 1)

Modelling the security property using Horn clauses

We consider secrecy as a reachability (accessibility) property.

Is $\mathcal{C}_{att} + \mathcal{C}_{prot} + \neg\text{att}(s)$ satisfiable or not?

Denning Sacco protocol

$\text{att}(sk_I)$	initial knowledge
$\text{att}(\text{pk}(sk_I))$	using attacker rules
$\text{att}(\text{aenc}(\text{sign}(k[sk_I], sk_A), \text{pk}(sk_I)))$	using protocol (rule 1)
$\text{att}(\text{aenc}(\text{sign}(k[sk_I], sk_A), \text{pk}(sk_B))$	using attacker rules and $\text{att}(\text{pk}(sk_B)$ (initial knowledge)

Modelling the security property using Horn clauses

We consider secrecy as a reachability (accessibility) property.

Is $\mathcal{C}_{att} + \mathcal{C}_{prot} + \neg\text{att}(s)$ satisfiable or not?

Denning Sacco protocol

$\text{att}(sk_I)$	initial knowledge
$\text{att}(\text{pk}(sk_I))$	using attacker rules
$\text{att}(\text{aenc}(\text{sign}(k[sk_I], sk_A), \text{pk}(sk_I)))$	using protocol (rule 1)
$\text{att}(\text{aenc}(\text{sign}(k[sk_I], sk_A), \text{pk}(sk_B))$	using attacker rules and $\text{att}(\text{pk}(sk_B)$ (initial knowledge)
$\text{att}(\text{senc}(s, k[sk_I]))$	using protocol (rule 2)

Modelling the security property using Horn clauses

We consider secrecy as a reachability (accessibility) property.

Is $\mathcal{C}_{att} + \mathcal{C}_{prot} + \neg\text{att}(s)$ satisfiable or not?

Denning Sacco protocol

$\text{att}(sk_I)$	initial knowledge
$\text{att}(\text{pk}(sk_I))$	using attacker rules
$\text{att}(\text{aenc}(\text{sign}(k[sk_I], sk_A), \text{pk}(sk_I)))$	using protocol (rule 1)
$\text{att}(\text{aenc}(\text{sign}(k[sk_I], sk_A), \text{pk}(sk_B))$	using attacker rules and $\text{att}(\text{pk}(sk_B)$ (initial knowledge)
$\text{att}(\text{senc}(s, k[sk_I]))$	using protocol (rule 2)
$\text{att}(k[sk_I])$	using attacker rules

Modelling the security property using Horn clauses

We consider secrecy as a reachability (accessibility) property.

Is $\mathcal{C}_{att} + \mathcal{C}_{prot} + \neg\text{att}(s)$ satisfiable or not?

Denning Sacco protocol

$\text{att}(sk_I)$	initial knowledge
$\text{att}(\text{pk}(sk_I))$	using attacker rules
$\text{att}(\text{aenc}(\text{sign}(k[sk_I], sk_A), \text{pk}(sk_I)))$	using protocol (rule 1)
$\text{att}(\text{aenc}(\text{sign}(k[sk_I], sk_A), \text{pk}(sk_B))$	using attacker rules and $\text{att}(\text{pk}(sk_B)$ (initial knowledge)
$\text{att}(\text{senc}(s, k[sk_I]))$	using protocol (rule 2)
$\text{att}(k[sk_I])$	using attacker rules
$\text{att}(s)$	using decryption

Modelling the security property using Horn clauses

We consider secrecy as a reachability (accessibility) property.

Is $\mathcal{C}_{att} + \mathcal{C}_{prot} + \neg\text{att}(s)$ satisfiable or not?

Denning Sacco protocol

$\text{att}(sk_I)$	initial knowledge
$\text{att}(\text{pk}(sk_I))$	using attacker rules
$\text{att}(\text{aenc}(\text{sign}(k[sk_I], sk_A), \text{pk}(sk_I)))$	using protocol (rule 1)
$\text{att}(\text{aenc}(\text{sign}(k[sk_I], sk_A), \text{pk}(sk_B))$	using attacker rules and $\text{att}(\text{pk}(sk_B)$ (initial knowledge)
$\text{att}(\text{senc}(s, k[sk_I]))$	using protocol (rule 2)
$\text{att}(k[sk_I])$	using attacker rules
$\text{att}(s)$	using decryption

Contradiction with $\neg\text{att}(s)$!

$\longrightarrow$ This set of clauses in **not** satisfiable.

How to decide satisfiability?

$\longrightarrow$ using resolution techniques

$$\frac{H \Rightarrow \mathsf{att}(u) \quad \mathsf{att}(v), H' \Rightarrow C}{(H, H' \Rightarrow C)\theta}\ \theta = \mathsf{mgu}(u, v) \quad \text{Resolution}$$

How to decide satisfiability?

$\longrightarrow$ using resolution techniques

$$\frac{H \Rightarrow \mathsf{att}(u) \quad \mathsf{att}(v), H' \Rightarrow C}{(H, H' \Rightarrow C)\theta}\ \theta = \mathsf{mgu}(u, v) \quad \text{Resolution}$$

Example

$$\frac{\Rightarrow \mathsf{att}(\mathsf{pk}(sk_I)) \quad \mathsf{att}(\mathsf{pk}(x)) \Rightarrow \mathsf{att}(\mathsf{aenc}(\mathsf{sign}(k[x], sk_A), \mathsf{pk}(x)))}{\Rightarrow \mathsf{att}(\mathsf{aenc}(\mathsf{sign}(k[sk_I], sk_A), pk(sk_I)))}\ \theta = \{x \mapsto sk_I\}$$

How to decide satisfiability?

$\longrightarrow$ using resolution techniques

$$\frac{H \Rightarrow \mathsf{att}(u) \quad \mathsf{att}(v), H' \Rightarrow C}{(H, H' \Rightarrow C)\theta}\ \theta = \mathsf{mgu}(u, v) \quad \text{Resolution}$$

Example

$$\frac{\Rightarrow \mathsf{att}(\mathsf{pk}(sk_I)) \quad \mathsf{att}(\mathsf{pk}(x)) \Rightarrow \mathsf{att}(\mathsf{aenc}(\mathsf{sign}(k[x], sk_A), \mathsf{pk}(x)))}{\Rightarrow \mathsf{att}(\mathsf{aenc}(\mathsf{sign}(k[sk_I], sk_A), pk(sk_I)))}\ \theta = \{x \mapsto sk_I\}$$

Theorem (soundness and completeness)

Resolution sound and refutationally complete, i.e. a set of Horn clauses $\mathcal{C}$ is not satisfiable if and only if $\square$ (the empty clause) can be obtained from $\mathcal{C}$ by using the resolution rule.

Exercises

Consider the Horn clauses given on the previous slides to model the Denning Sacco protocol.

Exercise

Exhibit an infinite derivation (using resolution).

Exercise

Apply binary resolution to derive the empty clause.

ProVerif

ProVerif implements a resolution strategy well-adapted to protocols.

Approximation of the translation in Horn clauses:

- the freshness of nonces is partially modeled;
- the number of times a message appears is ignored, only the fact that is has appeared is taken into account;
- the state of the principals is not fully modeled.

$\longrightarrow$ These approximations are keys for an efficient verification.

Experimental results

$\longrightarrow$ ProVerif works well in practice.

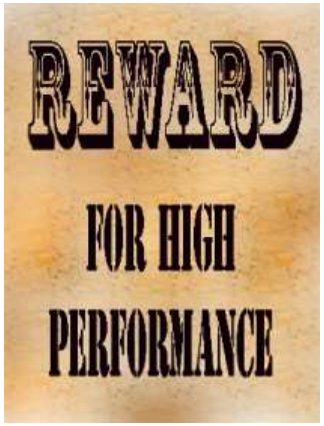
Protocol	Result	ms
Needham-Schroeder shared key	Attack	52
Needham-Schroeder shared key corrected	Secure	109
Denning-Sacco	Attack	6
Denning-Sacco corrected	Secure	7
Otway-Rees	Secure	10
Otway-Rees, variant of Paulson98	Attack	12
Yahalom	Secure	10
Simpler Yahalom	Secure	11
Main mode of Skeme	Secure	23

Pentium III, 1 GHz.

Challenge (for this afternoon)

Would you be able to find the attack on the well-known Needham-Schroeder protocol?

$$
\begin{aligned}
A \to B: &\quad \{A, N_a\}_{\mathsf{pub}(B)} \\
B \to A: &\quad \{N_a, N_b\}_{\mathsf{pub}(A)} \\
A \to B: &\quad \{N_b\}_{\mathsf{pub}(B)}
\end{aligned}
$$



See you this afternoon!