Verification of security protocols: from confidentiality to privacy

Stéphanie Delaune

Univ Rennes, CNRS, IRISA, France

Thursday, June 28th, 2018

Research at IRISA (Rennes)

⟶ 800 members (among which about 400 reasearchers)

EMSEC team

Embedded Security & Cryptography

⟶ 7 permanent researchers, 12 PhD students, and 2 post-docs

P. Derbez, G. Avoine, A. Roux-Langlois, B. Kordy, P.-A. Fouque
+ C. Maurice and myself.

Advertisement

POPSTAR ERC Project (2017-2022)
Reasoning about Physical properties
Of security Protocols
with an Application To contactless Systems

https://project.inria.fr/popstar/

Regular job offers:

- PhD positions and Post-doc positions;
- One research associate position (up to 5 years).

⟶ contact me: stephanie.delaune@irisa.fr

Cryptographic protocols everywhere !

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

A variety of security properties

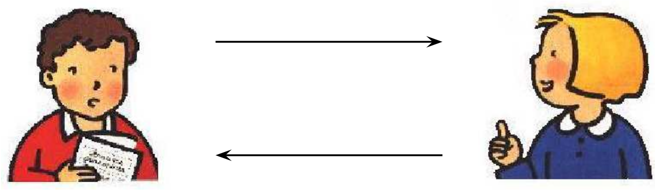
- **Secrecy**: May an intruder learn some secret message exchanged between two honest participants?
- Authentication: Is the agent Alice really talking to Bob?

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.
- ...

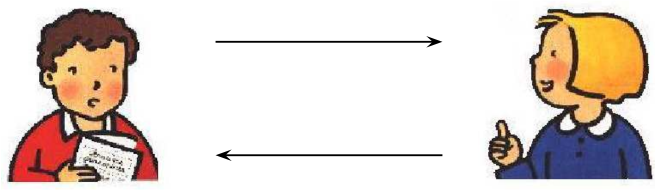
How does a cryptographic protocol work (or not)?

Protocol: small programs explaining how to exchange messages



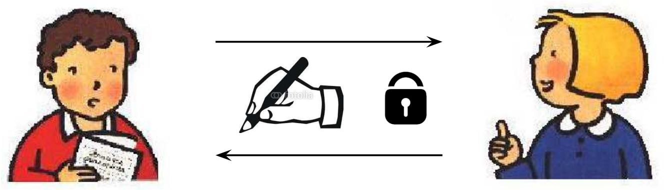
How does a cryptographic protocol work (or not)?

Protocol: small programs explaining how to exchange messages



How does a cryptographic protocol work (or not)?

Protocol: small programs explaining how to exchange messages

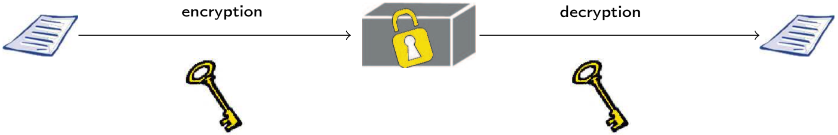


Cryptographic: make use of cryptographic primitives

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

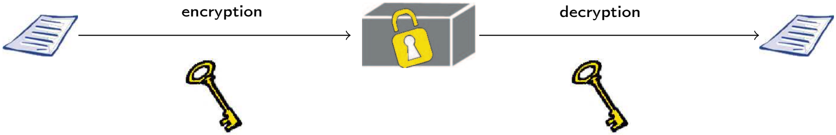
What is a symmetric encryption scheme?

Symmetric encryption

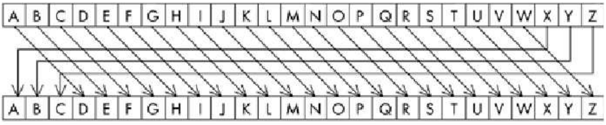


What is a symmetric encryption scheme?

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

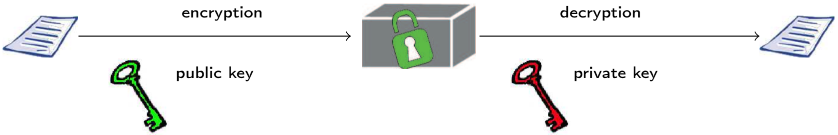
A bit of history

- 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.

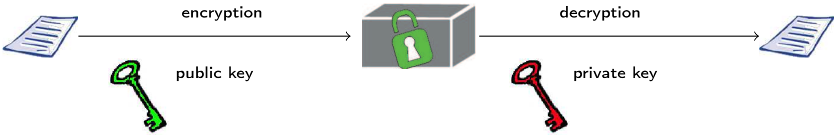
What is an asymmetric encryption scheme?

Asymmetric encryption



What is an asymmetric encryption scheme?

Asymmetric encryption



Examples:

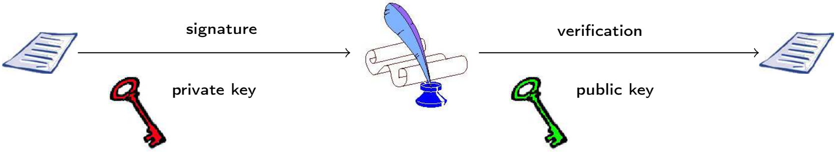
- 1976: first system published by W. Diffie, and M. Hellman,
- 1977: RSA system published by R. Rivest, A. Shamir, and L. Adleman.

$\longrightarrow$ 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,
  ↪ replay attack, man-in-the middle attack, ...
- subtle and hard to detect by "eyeballing" the protocol

⟶ 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)

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

Is the Denning Sacco protocol a good key exchange protocol?

Example: Denning Sacco protocol (1981)

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

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

Example: Denning Sacco protocol (1981)

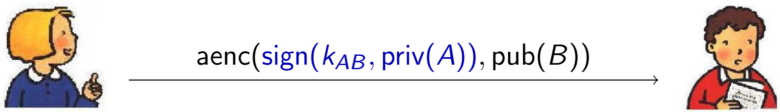
$\xrightarrow{\text{aenc}(\text{sign}(k_{AB}, \text{priv}(A)), \text{pub}(B))}$

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

Description of a possible attack:

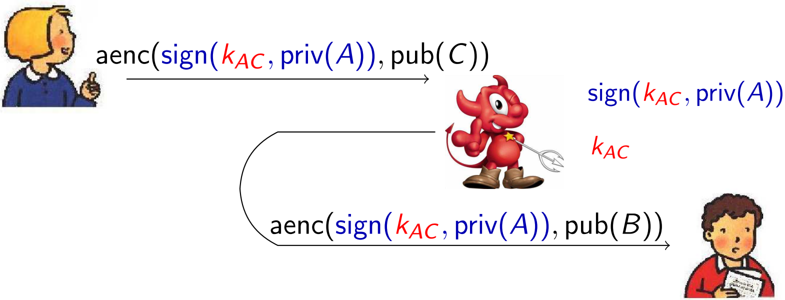
$\xrightarrow{\text{aenc}(\text{sign}(k_{AC}, \text{priv}(A)), \text{pub}(C))}$

Example: Denning Sacco protocol (1981)



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

Description of a possible attack:



Exercise

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

Version 1

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

Version 2

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

Which version would you prefer to use?

Exercise

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

Version 1

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

Version 2

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

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.

⟶ 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
6417406442524165008583957746445088405009430865999

$\longrightarrow$ 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)

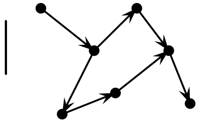
1. a logical flaw that allows a man in the middle attacker to downgrade connections from 'strong' RSA to 'export-grade' RSA;
2. breaking encryption via factorisation of such a key can be easily done.

⟶ '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?

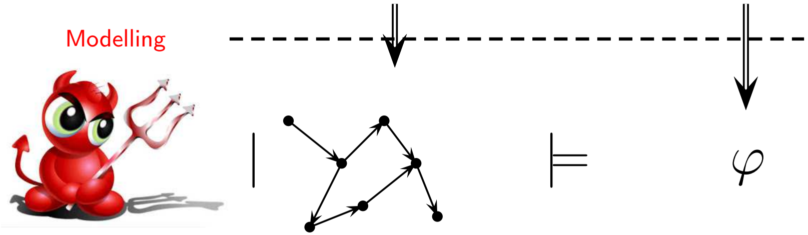
Modelling



$\models \varphi$

This talk: formal methods for protocol verification

Does the protocol *satisfy* a security property?



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.
⟶ 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
		$\text{in}(c, x).P$	input
		$\text{out}(c, u).P$	output
		$\text{if } u = v \text{ then } P \text{ else } Q$	conditional
		$P \mid Q$	parallel composition
		$!P$	replication
		$\text{new } n.P$	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 $\mathcal{F}$.

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

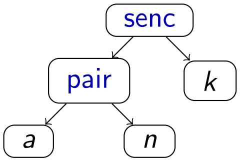
Messages as terms

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

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

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

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



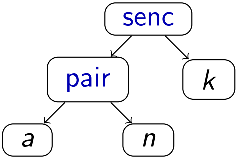
Messages as terms

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

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

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.

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

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

Semantics

Semantics $\rightarrow$:

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

Semantics

Semantics $\rightarrow$:

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

closed by

- structural equivalence ($\equiv$):

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

- application of evaluation contexts:

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

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

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

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

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

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

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)

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

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)

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

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

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

Alice and Bob as processes:

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

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

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

Alice and Bob as processes:

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

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

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)

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

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

We consider the following scenario:

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

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

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

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

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

We consider the following scenario:

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

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

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

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

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

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

We consider the following scenario:

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

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

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

$\longrightarrow$ 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 $A \mid P \to^* Q$, we have that Q is not of the form $C[\mathsf{out}(c, s).Q']$ with c public.

Security properties - confidentiality

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

For all processes A such that $A \mid P \to^* Q$, 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, ...

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 $C[\text{out}(c, s).Q']$ 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, ...

$\longrightarrow$ Going back to the Denning Sacco protocol

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]

⟶ 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.

⟶ 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`

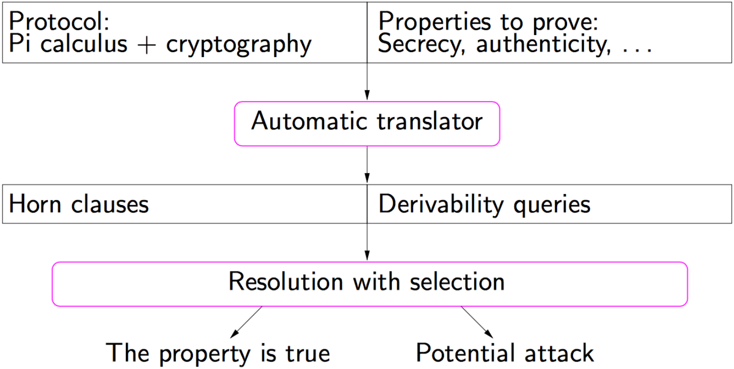
Advantages

- 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

$$
\begin{array}{rcl}
\mathrm{att}(x) & \Rightarrow & \mathrm{att}(\mathrm{pk}(x)) \\
\mathrm{att}(x),\ \mathrm{att}(\mathrm{pk}(y)) & \Rightarrow & \mathrm{att}(\mathrm{aenc}(x, \mathrm{pk}(y))) \\
\mathrm{att}((\mathrm{aenc}(x, \mathrm{pk}(y))),\ \mathrm{att}(y) & \Rightarrow & \mathrm{att}(x)
\end{array}
$$

Signature

$$
\begin{array}{rcl}
\mathrm{att}(x),\ \mathrm{att}(y) & \Rightarrow & \mathrm{att}(\mathrm{sign}(x, y)) \\
\mathrm{att}(\mathrm{sign}(x, y)) & \Rightarrow & \mathrm{att}(x)
\end{array}
$$

Symmetric encryption

$$
\begin{array}{rcl}
\mathrm{att}(x),\ \mathrm{att}(y) & \Rightarrow & \mathrm{att}(\mathrm{senc}(x, y)) \\
\mathrm{att}((\mathrm{senc}(x, y)),\ \mathrm{att}(y) & \Rightarrow & \mathrm{att}(x)
\end{array}
$$

Initial knowledge

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

Modelling the protococol using Horn clauses

Denning-Sacco protocol ...

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

... using Horn clauses

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

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

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

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

Modelling the protococol using Horn clauses

Denning-Sacco protocol ...

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

... using Horn clauses

- A talks with any principal represented by its public key pk(x).

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

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

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

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

Modelling the security property using Horn clauses

We consider secrecy as a reachability (accessibility) property.

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

Modelling the security property using Horn clauses

We consider secrecy as a reachability (accessibility) property.

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

Denning Sacco protocol

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

Modelling the security property using Horn clauses

We consider secrecy as a reachability (accessibility) property.

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

Denning Sacco protocol

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

Modelling the security property using Horn clauses

We consider secrecy as a reachability (accessibility) property.

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

Denning Sacco protocol

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

Modelling the security property using Horn clauses

We consider secrecy as a reachability (accessibility) property.

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

Denning Sacco protocol

$\mathrm{att}(sk_I)$	initial knowledge
$\mathrm{att}(\mathrm{pk}(sk_I))$	using attacker rules
$\mathrm{att}(\mathrm{aenc}(\mathrm{sign}(k[sk_I], sk_A), \mathrm{pk}(sk_I)))$	using protocol (rule 1)
$\mathrm{att}(\mathrm{aenc}(\mathrm{sign}(k[sk_I], sk_A), \mathrm{pk}(sk_B))$	using attacker rules and att(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 $\mathcal{C}_{att} + \mathcal{C}_{prot} + \neg \mathrm{att}(s)$ satisfiable or not?

Denning Sacco protocol

$\mathrm{att}(sk_I)$	initial knowledge
$\mathrm{att}(\mathrm{pk}(sk_I))$	using attacker rules
$\mathrm{att}(\mathrm{aenc}(\mathrm{sign}(k[sk_I], sk_A), \mathrm{pk}(sk_I)))$	using protocol (rule 1)
$\mathrm{att}(\mathrm{aenc}(\mathrm{sign}(k[sk_I], sk_A), \mathrm{pk}(sk_B))$	using attacker rules and att(pk(sk_B) (initial knowledge)
$\mathrm{att}(\mathrm{senc}(s, k[sk_I]))$	using protocol (rule 2)
$\mathrm{att}(k[sk_I])$	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 $\mathcal{C}_{att} + \mathcal{C}_{prot} + \neg \mathsf{att}(s)$ satisfiable or not?

Denning Sacco protocol

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

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

$\longrightarrow$ 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

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

Example

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

Theorem (soundness and completeness)

Resolution is 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 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.

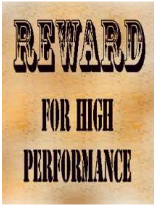
Protocol	Result	ms
Needham-Schroeder shared key	Attack	52
Needham-Schroeder shared key corrected	Secure	109
Denning-Sacco	Attack	6
Denning-Sacco corrected	Secure	7
Otway-Rees	Secure	10
Otway-Rees, variant of Paulson98	Attack	12
Yahalom	Secure	10
Simpler Yahalom	Secure	11
Main mode of Skeme	Secure	23

Pentium III, 1 GHz.

Challenge (to discuss during the break)

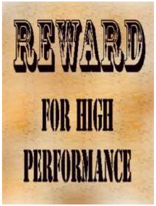
Would you be able to find the attack on the well-known Needham-Schroeder protocol?

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



Challenge (to discuss during the break)

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



$$\begin{array}{ll} A \rightarrow B: & \{A, N_a\}_{\mathrm{pub}(B)} \\ B \rightarrow A: & \{N_a, N_b\}_{\mathrm{pub}(A)} \\ A \rightarrow B: & \{N_b\}_{\mathrm{pub}(B)} \end{array}$$

Questions

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

Verification of security protocols: from confidentiality to privacy

Stéphanie Delaune

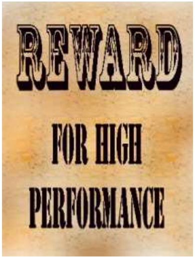
Univ Rennes, CNRS, IRISA, France

Thursday, June 28th, 2018

Challenge (1/2)

Would you be able to find the attack on the well-known Needham-Schroeder protocol (1978)?

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



Questions

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

Challenge (2/2)

An attack has been found 17 years after the publication of this protocol !

Man in the middle attack due to G. Lowe 1995

- involving 2 sessions in parallel,
- an honest agent has to initiate a session with C.

Fixed version of the protocol

$$
\begin{array}{lll}
\mathsf{A} \rightarrow \mathsf{B} & : & \{A, N_a\}_{\mathsf{pub}(B)} \\
\mathsf{B} \rightarrow \mathsf{A} & : & \{N_a, N_b, B\}_{\mathsf{pub}(A)} \\
\mathsf{A} \rightarrow \mathsf{B} & : & \{N_b\}_{\mathsf{pub}(B)}
\end{array}
$$

$\longrightarrow$ the responder's identity has been added to the second message

Security protocols everywhere !

It becomes more and more important to protect our privacy.

Electronic passport

An e-passport is a passport with an RFID tag embedded in it.

The RFID tag stores:

- the information printed on your passport;
- a JPEG copy of your picture;
- ...

The Basic Access Control (BAC) protocol is a key establishment protocol that has been designed to protect our personnal data, and to ensure unlinkability.

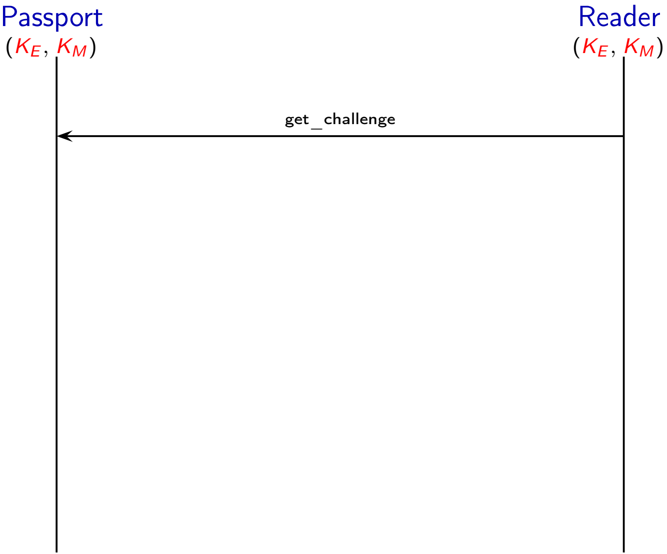
Unlinkability aims to ensure *that a user may make multiple uses of a service or resource without others being able to link these uses together.* [ISO/IEC standard 15408]

BAC protocol

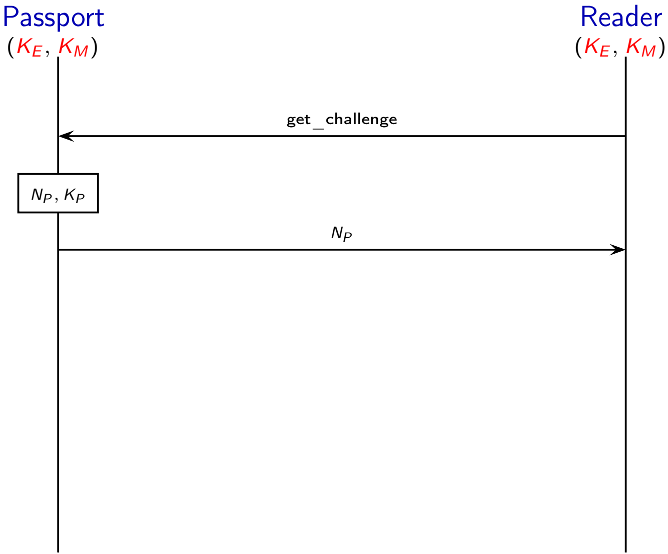
Passport
(K_E, K_M)

Reader
(K_E, K_M)

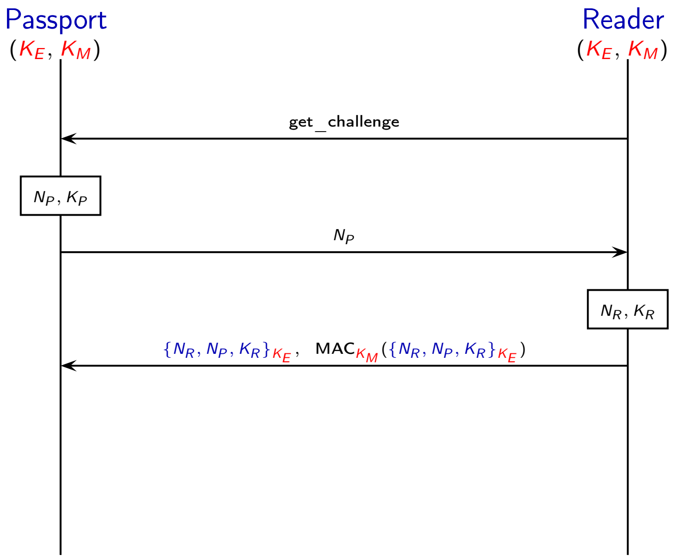
BAC protocol



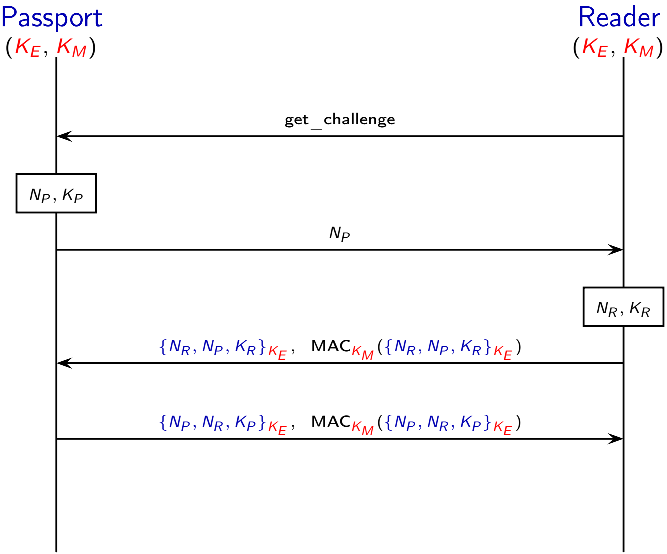
BAC protocol



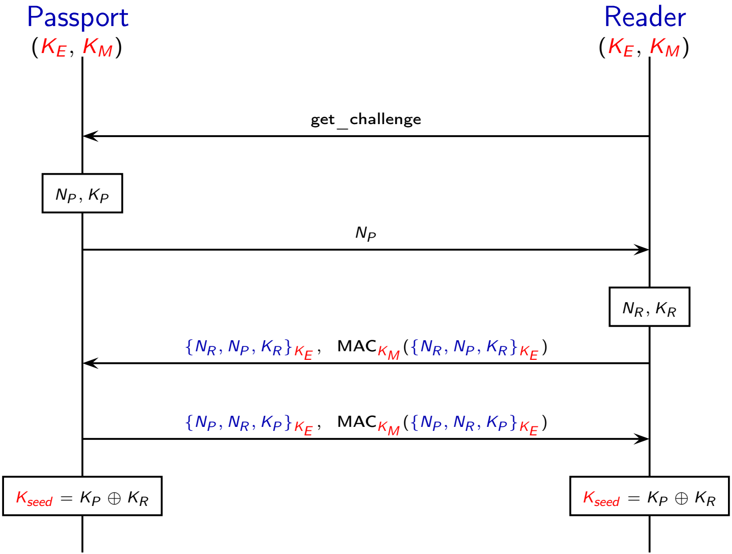
BAC protocol



BAC protocol



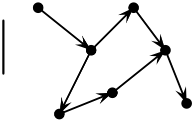
BAC protocol



A brief recap

Does the protocol *satisfy* a security property?

Modelling



$\models \varphi$

How can we check privacy-type security properties?

Part I

Modelling protocols, **security properties**

and the attacker

Messages as terms (on an example)

Nonces n_r, n_p, and keys k_r, k_p, k_e, k_m are modelled using names

Cryptographic primitives are modelled using function symbols

- encryption/decryption: senc/2, sdec/2
- concatenation/projections: $\langle\ ,\ \rangle/2$, $\mathsf{proj}_1/1$, $\mathsf{proj}_2/1$
- mac construction: mac/2

Properties of the primitives are modelled using an equational theory.

$$\mathsf{sdec}(\mathsf{senc}(x, y), y) = x, \quad \mathsf{proj}_1(\langle x, y\rangle) = x, \quad \mathsf{proj}_2(\langle x, y\rangle) = y.$$

Protocols as processes (on an example)

$$P \to R: \quad N_p$$
$$R \to P: \quad \{N_R, N_P, K_R\}_{K_E}, \;\; \mathrm{MAC}_{K_M}(\{N_R, N_P, K_R\}_{K_E})$$
$$P \to R: \quad \{N_P, N_R, K_P\}_{K_E}, \;\; \mathrm{MAC}_{K_M}(\{N_P, N_R, K_P\}_{K_E})$$

Modelling Passport's role

$$
\begin{aligned}
P_{\mathrm{BAC}}(k_E, k_M) \;=\; & \mathsf{new}\ n_P.\mathsf{new}\ k_P.\mathsf{out}(n_P).\mathsf{in}(\langle z_E, z_M\rangle).\\
& \mathtt{if}\ z_M = \mathsf{mac}(z_E, k_M)\ \mathtt{then}\ \mathtt{if}\ n_P = \mathsf{proj}_1(\mathsf{proj}_2(\mathsf{sdec}(z_E, k_E)))\\
& \qquad\qquad\qquad\qquad\qquad \mathtt{then}\ \mathsf{out}(\langle m, \mathsf{mac}(m, k_M)\rangle)\\
& \qquad\qquad\qquad\qquad\qquad \mathtt{else}\ 0\\
& \qquad\qquad\qquad\qquad \mathtt{else}\ 0
\end{aligned}
$$

where $m = \mathsf{senc}(\langle n_P, \langle \mathsf{proj}_1(z_E), k_P\rangle\rangle, k_E)$.

What does unlinkability mean?

Informally, an attacker can not observe the difference between the two following situations:

1. a situation where the same passport may be used twice (or even more);
2. a situation where each passport is used at most once.

What does unlinkability mean?

Informally, an attacker can not observe the difference between the two following situations:

1. a situation where the same passport may be used twice (or even more);
2. a situation where each passport is used at most once.

More formally,

$$!\mathrm{new}\ ke.\mathrm{new}\ km.(!P_{\mathrm{BAC}} \mid !R_{\mathrm{BAC}}) \overset{?}{\approx} !\mathrm{new}\ ke.\mathrm{new}\ km.(P_{\mathrm{BAC}} \mid !R_{\mathrm{BAC}})$$

↑ many sessions for each passport

↑ only one session for each passport

(we still have to formalize the notion of equivalence)

Security properties - privacy

Privacy-type properties are modelled relying on testing equivalence.

Security properties - privacy

Privacy-type properties are modelled relying on testing equivalence.

Testing equivalence between P and Q, denoted $P \approx Q$

for all processes A, we have that:

$$(A \mid P) \Downarrow_c \text{ if, and only if, } (A \mid Q) \Downarrow_c$$

where $R \Downarrow_c$ means that R can evolve and emits on public channel c.

Security properties - privacy

Privacy-type properties are modelled relying on testing equivalence.

Testing equivalence between P and Q, denoted $P \approx Q$
for all processes A, we have that:

$$(A \mid P) \Downarrow_c \text{ if, and only if, } (A \mid Q) \Downarrow_c$$

where $R \Downarrow_c$ means that R can evolve and emits on public channel c.

Exercise 1: $\mathsf{out}(a, yes) \stackrel{?}{\approx} \mathsf{out}(a, no)$

Security properties - privacy

Privacy-type properties are modelled relying on testing equivalence.

Testing equivalence between P and Q, denoted $P \approx Q$

for all processes A, we have that:

$$(A \mid P) \Downarrow_c \text{ if, and only if, } (A \mid Q) \Downarrow_c$$

where $R \Downarrow_c$ means that R can evolve and emits on public channel c.

Exercise 1: $\quad \mathsf{out}(a, \mathit{yes}) \not\approx \mathsf{out}(a, \mathit{no})$

$$\longrightarrow \quad A = \mathsf{in}(a, x).\mathsf{if}\ x = \mathit{yes}\ \mathsf{then}\ \mathsf{out}(c, \mathit{ok})$$

Security properties - privacy

Privacy-type properties are modelled relying on testing equivalence.

Testing equivalence between P and Q, denoted $P \approx Q$

for all processes A, we have that:

$$(A \mid P) \Downarrow_c \text{ if, and only if, } (A \mid Q) \Downarrow_c$$

where $R \Downarrow_c$ means that R can evolve and emits on public channel c.

Exercise 2: k and k' are known to the attacker

$$\text{new } s.\text{out}(a, \text{senc}(s, k)).\text{out}(a, \text{senc}(s, k'))$$
$$\overset{?}{\approx}$$
$$\text{new } s, s'.\text{out}(a, \text{senc}(s, k)).\text{out}(a, \text{senc}(s', k'))$$

Security properties - privacy

Privacy-type properties are modelled relying on testing equivalence.

Testing equivalence between P and Q, denoted $P \approx Q$

for all processes A, we have that:

$$(A \mid P) \Downarrow_c \text{ if, and only if, } (A \mid Q) \Downarrow_c$$

where $R \Downarrow_c$ means that R can evolve and emits on public channel c.

Exercise 2: k and k' are known to the attacker

$$\text{new}\, s.\text{out}(a, \text{senc}(s, k)).\text{out}(a, \text{senc}(s, k'))$$

$$\not\approx$$

$$\text{new}\, s, s'.\text{out}(a, \text{senc}(s, k)).\text{out}(a, \text{senc}(s', k'))$$

$$\longrightarrow \text{in}(a, x).\text{in}(a, y).\text{if } (\text{sdec}(x, k) = \text{sdec}(y, k')) \text{ then out}(c, ok)$$

Security properties - privacy

Privacy-type properties are modelled relying on testing equivalence.

Testing equivalence between P and Q, denoted $P \approx Q$
for all processes A, we have that:

$$(A \mid P) \Downarrow_c \text{ if, and only if, } (A \mid Q) \Downarrow_c$$

where $R \Downarrow_c$ means that R can evolve and emits on public channel c.

Exercise 3: Are the two following processes in testing equivalence?

$$\text{new } s.\text{out}(a, s) \overset{?}{\approx} \text{new } s.\text{new } k.\text{out}(a, \text{senc}(s, k))$$

Some other equivalence-based security properties

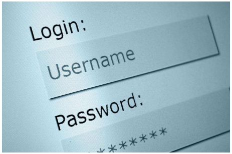
The notion of testing equivalence can be used to express:

Vote privacy

the fact that a particular voted in a particular way is not revealed to anyone

Strong secrecy

the fact that an adversary cannot see any difference when the value of the secret changes
$\longrightarrow$ stronger than the notion of secrecy as non-deducibility.



Guessing attack

the fact that an adversary can not learn the value of passwords even if he knows that they have been choosen in a particular dictionary.

Part II

Designing verification algorithms

privacy-type properties

State of the art for testing equivalence (no !)

for analysing testing equivalence

bounded number of sessions

State of the art for testing equivalence (no !)

for analysing testing equivalence
bounded number of sessions

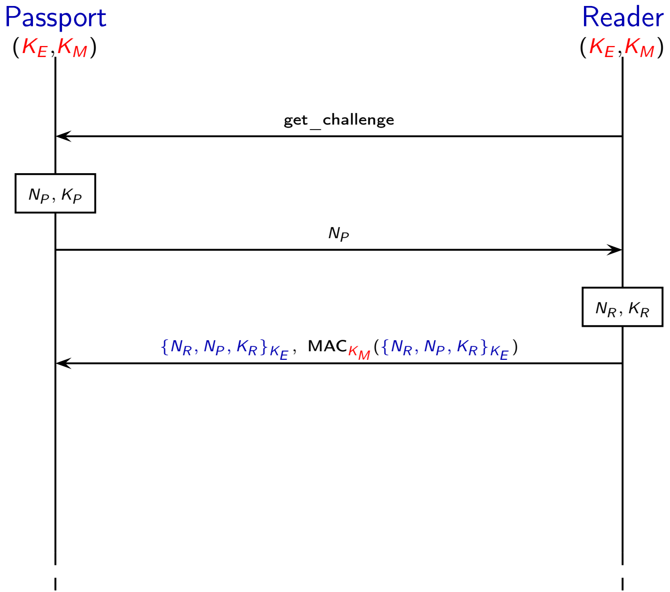
Some important results:

- A decision procedure implemented in the tool Apte:
  non-trivial else branches, private channels, and
  non-deterministic choice, a fixed set of primitives
  [Cheval, Comon & D., 11]
- A procedure implemented in the tool Akiss:
  no else branches, but a larger class of primitives
  [Chadha et al, 12]

⟶ A decision procedure implemented in the tool DEEPSEC
[Cheval, Kremer & Rakotonirina, 2018]

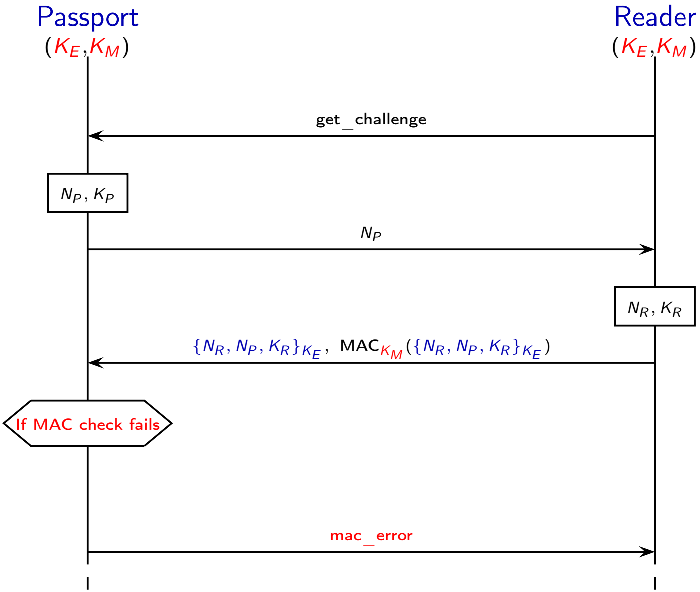
French electronic passport

⟶ the passport must reply to all received messages.



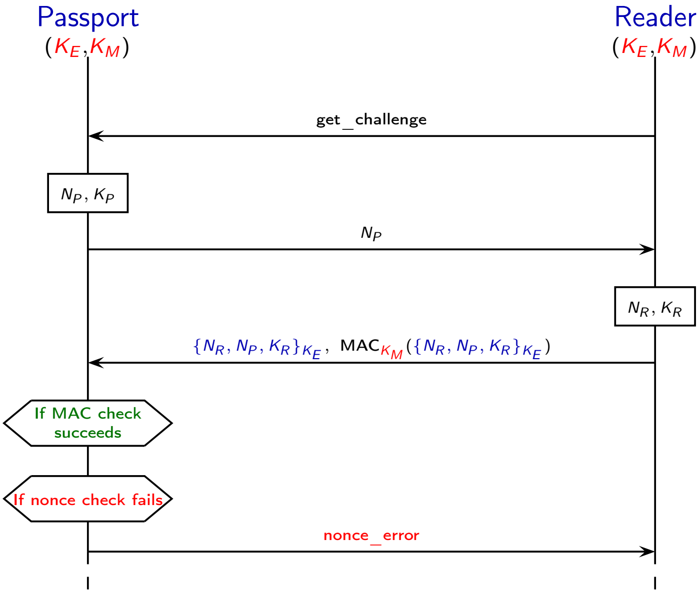
French electronic passport

$\longrightarrow$ the passport must reply to all received messages.



French electronic passport

$\longrightarrow$ the passport must reply to all received messages.



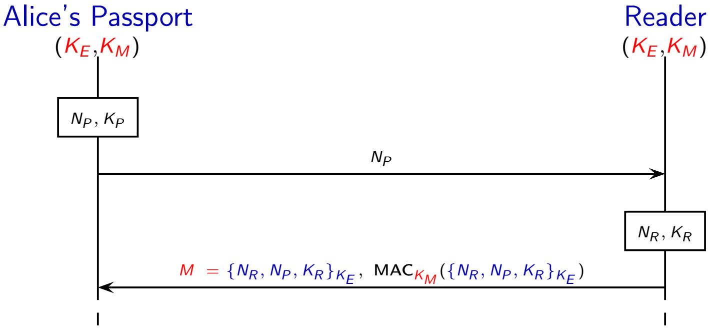
An attack on the French passport [Chothia & Smirnov, 10]

An attacker can track a French passport, provided he has once witnessed a successful authentication.

An attack on the French passport [Chothia & Smirnov, 10]

An attacker can track a French passport, provided he has once witnessed a successful authentication.

Part 1 of the attack. The attacker eavesdropes on Alice using her passport and records message M.

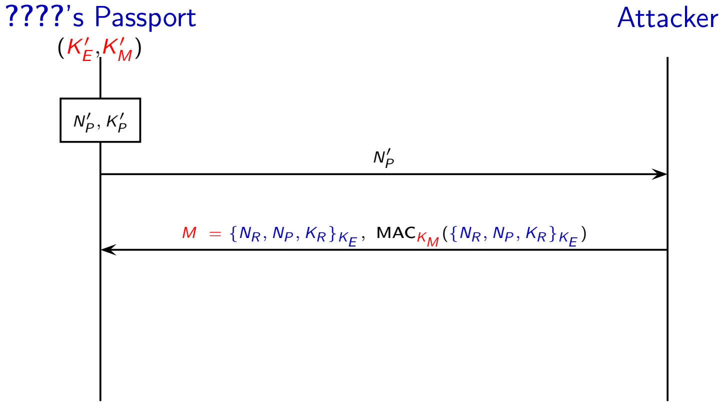


An attack on the French passport [Chothia & Smirnov, 10]

An attacker can track a French passport, provided he has once witnessed a successful authentication.

Part 2 of the attack.

The attacker replays M and checks the error code he receives.

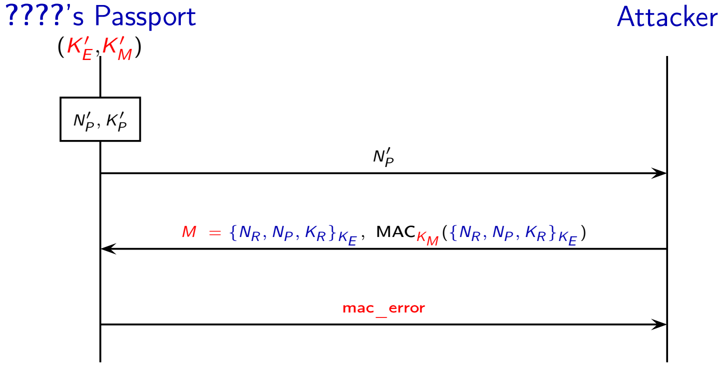


An attack on the French passport [Chothia & Smirnov, 10]

An attacker can track a French passport, provided he has once witnessed a successful authentication.

Part 2 of the attack.

The attacker replays M and checks the error code he receives.



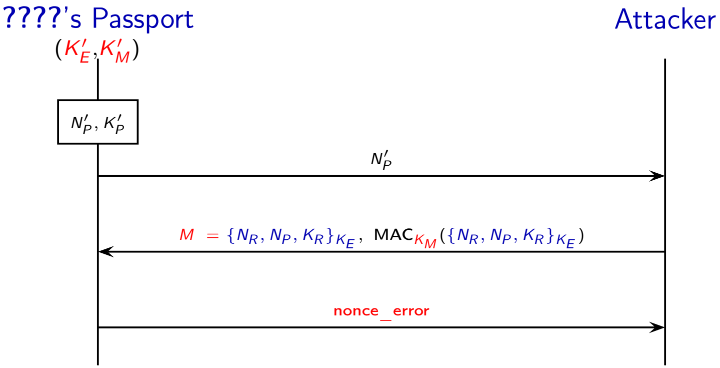
$\Longrightarrow$ MAC check failed $\Longrightarrow$ $K'_M \neq K_M$ $\Longrightarrow$ **????** is not Alice

An attack on the French passport [Chothia & Smirnov, 10]

An attacker can track a French passport, provided he has once witnessed a successful authentication.

Part 2 of the attack.

The attacker replays M and checks the error code he receives.



$\Longrightarrow$ MAC check succeeded $\Longrightarrow$ $K'_M = K_M$ $\Longrightarrow$ **????** is Alice

State of the art for testing equivalence (with !)

for analysing testing equivalence

unbounded number of sessions

State of the art for testing equivalence (with !)

for analysing testing equivalence
unbounded number of sessions

- undecidable in general even for some fragment for which confidentiality is decidable [Chrétien, Cortier & D., 13]
- some recent decidability results for some restricted fragment *e.g.* tagged protocols, no nonces, a particular set of primitives ... [Chrétien, Cortier & D., Icalp'13, Concur'14, CSF'15]
- some existing verification tools: ProVerif, Tamarin, ... for analysing the notion of diff-equivalence (stronger than testing equivalence) [Blanchet, Abadi & Fournet, 05] [Basin, Dreier & Sasse, 15]

None of these results is suitable to analyse vote-privacy, or unlinkability of the BAC protocol.

Diff-equivalence is often too strong in practice

Vote privacy
the fact that a particular voted in a particular way is not revealed to anyone

$$V_A(\textcolor{red}{yes}) \mid V_B(\textcolor{blue}{no}) \approx V_A(\textcolor{blue}{no}) \mid V_B(\textcolor{red}{yes})$$

$\longrightarrow$ ProSwapper extension [Blanchet & Smyth, 2016]

Diff-equivalence is often too strong in practice

Vote privacy
the fact that a particular voted in a particular way is not revealed to anyone

$$V_A(yes) \mid V_B(no) \approx V_A(no) \mid V_B(yes)$$

$\longrightarrow$ ProSwapper extension [Blanchet & Smyth, 2016]

Unlinkability a user may make multiple uses of a resource without other being able to link these uses together.

$$!\,\mathrm{new}\; k.!P \;\approx\; !\,\mathrm{new}\; k.P$$

$\longrightarrow$ UKANO extension [Hirschi, Baelde, & D, 2016]

UKANO extension (1/2) [Hirschi, Baelde, & D, 2016]

Provide a method to analyse unlinkability for a large class of 2 party protocols, and tool support for that.

UKANO extension (1/2) [Hirschi, Baelde, & D, 2016]

Provide a method to analyse unlinkability for a large class of 2 party protocols, and tool support for that.

On the theoretical side

2 reasonable conditions implying anonymity and unlinkability for a large class of 2 party protocols

On the practical side

- our conditions can be checked automatically using existing tools, and we provide tool support for that.
- new proofs and attacks on several RFID protocols.

$\longrightarrow$ first results published at Security & Privacy in 2016 extended since to deal with a larger class of processes

UKANO extension (2/2) – summary of our case studies

Protocol	FO	WA	unlinkability
Feldhofer	✓	✓	safe
Feldhofer variant (with !)	✓	✗	attack
Hash-Lock	✓	✓	safe
LAK (stateless)	–	✗	attack
Fixed LAK	✓	✓	safe
BAC	✓	✓	safe
BAC/PA/AA	✓	✓	safe
PACE (faillible dec)	–	✗	attack
PACE (as in [Bender et al, 09])	–	✗	attack
PACE	–	✗	attack
PACE with tags	✓	✓	safe
DAA sign	✓	✓	safe
DAA join	✓	✓	safe
abcdh (irma)	✓	✓	safe

Conclusion

To sum up

Cryptographic protocols are:

- difficult to design and analyse;
- particularly vulnerable to logical attacks.

Strong primitives are necessary ...

... **but this is not sufficient !**

To sum up

Cryptographic protocols are:

- difficult to design and analyse;
- particularly vulnerable to logical attacks.

It is important to ensure that
the protocols we are using every day work properly.

We now have automatic and powerful verification tools to analyse:

- classical security goals, *e.g.* secrecy and authentication;
- relatively small protocols;
- protocols that rely on standard cryptographic primitives.

Limitations of the symbolic approach

1. the algebraic properties of the primitives are abstracted away
   $\longrightarrow$ no guarantee if the protocol relies on an encryption that satisfies some additional properties (*e.g.* RSA, ElGamal)

2. only the specification is analysed and not the implementation
   $\longrightarrow$ most of the passports are actually linkable by a carefull analysis of time or message length.

   http://www.loria.fr/g̃londu/epassport/attaque-tailles.html

3. when considering a bounded number of sessions, not all scenario are checked
   $\longrightarrow$ no guarantee if the protocol is used one more time !

It remains a lot to do

- formal definitions of some sublte security properties
  $\longrightarrow$ receipt-freeness, coercion-resistance in e-voting
- algorithms (and tools!) for checking automatically trace equivalence for various cryptographic primitives;
  $\longrightarrow$ homomorphic encryption used in e-voting, exclusive-or used in RFID protocols
- more composition results
  $\longrightarrow$ Could we derive some security guarantees of the whole e-passport application from the analysis performed on each subprotocol?
- develop more fine-grained models (and tools) to take into account side channel attacks
  $\longrightarrow$ e.g. timing attacks

Questions ?