Symbolic verification of security protocols
Modelling and verifying unlinkability

Stéphanie DELAUNE, Univ Rennes, CNRS, IRISA

GT Méthodes Formelles pour la Sécurité

March 29th, 2023

⟶ joint work with D. Baelde, A. Debant, L. Hirschi, and S. Moreau

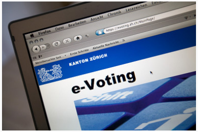
Cryptographic protocols everywhere !

- small programs designed to secure communication (*e.g.* secrecy, authentication, anonymity, ...)
- use cryptographic primitives (*e.g.* encryption, signature, ...)

Cryptographic protocols everywhere !

- small programs designed to secure communication (*e.g.* secrecy, authentication, anonymity, . . .)
- use cryptographic primitives (*e.g.* encryption, signature, ...)

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;
- ...

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]

How cryptographic protocols can be attacked?

How cryptographic protocols can be attacked?

Logical attacks

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

Example: A traceability attack on the BAC protocol

Security

Defects in e-passports allow real-time tracking

This threat brought to you by RFID

The register - Jan. 2010

How to verify the absence of logical flaws?

- dissect the protocol and test their resilience against well-known attacks;
  $\longrightarrow$ this is not sufficient !

How to verify the absence of logical flaws?

- dissect the protocol and test their resilience against well-known attacks;
  $\longrightarrow$ this is not sufficient !

- perform a manual security analysis
  $\longrightarrow$ this is error-prone !

How to verify the absence of logical flaws?

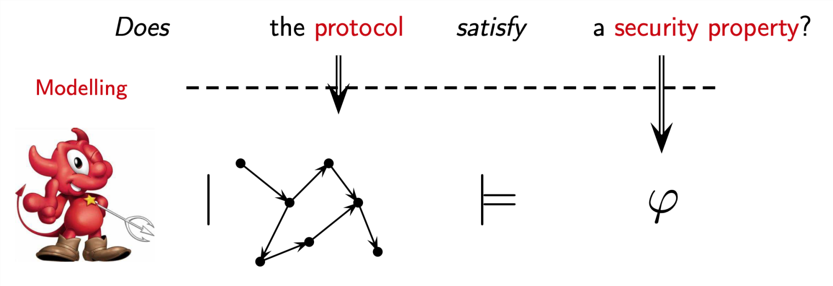
- dissect the protocol and test their resilience against well-known attacks;
  $\longrightarrow$ this is not sufficient !

- perform a manual security analysis
  $\longrightarrow$ this is error-prone !

Our approach

formal symbolic verification using automatic/interactive tools

Formal (symbolic) verification in a nutshell



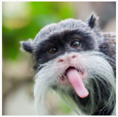
Two main tasks:

1. Modelling: protocols, security properties, and the attacker;
2. Verifying: designing verification algorithms and tools.

$\longrightarrow$ this talk: a focus on unlinkability

Some success stories (mostly related to reachability properties)

ProVerif [Blanchet, 01]

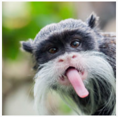


[Meier et al., 13]

Sapic^{+} [Cheval *et al.*, 22]

Some success stories (mostly related to reachability properties)

ProVerif [Blanchet, 01]



[Meier et al., 13]

Sapic$^+$ [Cheval *et al.*, 22]

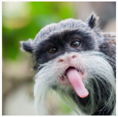
Verified models and reference implementations for TLS 1.3 [Bhargavan *et al.*, 17]

A formal security analysis of the EMV Standard using Tamarin (Break, Fix, and Verify) [Basin *et al.*, 2020]

A comprehensive, formal and automated analysis of the EDHOC protocol [Jacomme *et al*, 23]

Some success stories (mostly related to reachability properties)

ProVerif



[Blanchet, 01] [Meier et al., 13] [Cheval *et al.*, 22]

Verified models and reference implementations for TLS 1.3 [Bhargavan *et al.*, 17]

A formal security analysis of the EMV Standard using Tamarin (Break, Fix, and Verify) [Basin *et al.*, 2020]

A comprehensive, formal and automated analysis of the EDHOC protocol [Jacomme *et al*, 23]

What about unlinkability (in the symbolic setting)?

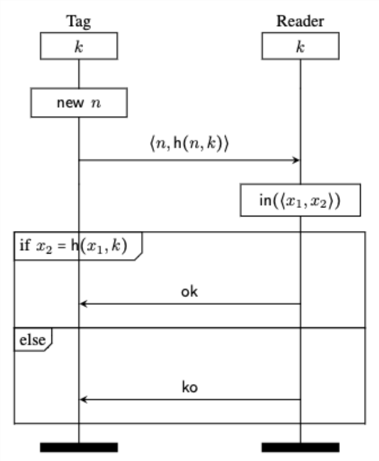
Actually, existing tools like ProVerif and Tamarin are not suitable to analyse unlinkability, and therefore few formal proofs exist in the unbounded setting.

- [Chatzikokolakis *et al.*, 2010]: sufficient conditions checkable using ProVerif that allows one to establish unlinkability for a simple class of protocols (single-step protocols).
  $\longrightarrow$ their notion of unlinkability is rather weak
- [Arapinis *et al.*, 2010]: a formal definition of unlinkability, and a manual proof of unlinkability for a fixed version of the e-passport protocol.
  $\longrightarrow$ this result is wrong
- [Bhargavan *et al.*, 2022]: a symbolic analysis of privacy for TLS 1.3 with Encrypted Client Hello
  $\longrightarrow$ several encodings tricks are used.

Part I

Modelling: protocols, the attacker, and unlinkability

Running example: Basic Hash protocol



- k is a long-term secret key shared between the tag and the reader;
- each tag has its own key k.

Protocols as processes

$\longrightarrow$ a programming language with constructs for concurrency and communication (applied-pi calculus [Abadi & Fournet, 01])

$$
\begin{array}{rcll}
P, Q & := & 0 & \text{null process} \\
 & | & \mathrm{in}(c, x); P & \text{input} \\
 & | & \mathrm{out}(c, M); P & \text{output} \\
 & | & \mathtt{new}\ n; P & \text{name generation} \\
 & | & \text{if } M = N \text{ then } P \text{ else } Q & \text{conditional} \\
 & | & !P & \text{replication} \\
 & | & (P \mid Q) & \text{parallel composition} \\
 & | & \ldots &
\end{array}
$$

Messages/Computations as terms

Terms are built over a set of names $\mathcal{N}$ (private), and function symbols Σ (public) equipped with an equational theory E.

Example:

$$\Sigma = \{\mathsf{senc}, \mathsf{sdec}\} \text{ with } \mathsf{E} = \{\mathsf{sdec}(\mathsf{senc}(x, y), y) = x\}.$$

Let $\Phi = \{w_1 \mapsto \mathsf{senc}(s, k); \mathsf{w}_2 \mapsto k\}$. $R = \mathsf{sdec}(\mathsf{w}_1, \mathsf{w}_2)$ is a recipe to compute s. Indeed, we have that $R\Phi =_{\mathsf{E}} s$.

Going back to Basic Hash

Mesages/Computations as terms

- $\Sigma = \{\mathsf{h},\ \langle\ \rangle,\ \mathsf{proj}_1,\ \mathsf{proj}_2\}$;
- $\mathsf{E} = \{\mathsf{proj}_1(\langle x_1, x_2\rangle) = x_1,\ \ \mathsf{proj}_2(\langle x_1, x_2\rangle) = x_2\}$.

Protocol as a process

- $\mathsf{T}(k) = \mathtt{new}\ n; \mathtt{out}(c, \langle n, \mathsf{h}(n, k)\rangle)$.
- $\mathsf{R}(k) = \mathtt{in}(c, y); \text{if } \mathsf{h}(\mathsf{proj}_1(y), k) = \mathsf{proj}_2(y)$
  $\text{then } \mathtt{out}(c, \mathsf{ok})$
  $\text{else } \mathtt{out}(c, \mathsf{ko})$.

Then, the whole system can be written as follows:

$$!\ \mathtt{new}\ k;\ (\ !\ \mathsf{R}(k)\ \mid\ !\ \mathsf{T}(k)\ )$$

Semantics (some selected rules)

Labelled transition system over configurations:

$$(\mathcal{P}; \Phi)$$

multiset of processes

frame = knowledge of the attacker

Semantics (some selected rules)

Labelled transition system over configurations:

$$(\mathcal{P}; \Phi)$$

multiset of processes

frame = knowledge of the attacker

$$\text{OUT} \quad (\{\text{out}(c, M); P\} \uplus \mathcal{P}; \Phi) \xrightarrow{\text{out}(c, w_i)} (\{P\} \uplus \mathcal{P}; \Phi \cup \{w_i \mapsto M\})$$

with $i = |\Phi|$

$$\text{THEN} \quad (\{\text{if } M_1 = M_2 \text{ then } P \text{ else } Q\} \uplus \mathcal{P}; \Phi) \xrightarrow{\tau} (\{P\} \uplus \mathcal{P}; \Phi)$$

when $M_1 =_{\mathsf{E}} M_2$

$$\text{IN} \quad (\{\text{in}(c, x); P\} \uplus \mathcal{P}; \Phi) \xrightarrow{\text{in}(c, R)} (\{P\{x \mapsto R\Phi\}\} \uplus \mathcal{P}; \Phi)$$

...

Trace equivalence

Trace equivalence between configurations: $K \approx_t K'$.

For any execution trace $K \xrightarrow{\mathsf{tr}} (\mathcal{P}; \Phi)$ there exists an execution $K' \xrightarrow{\mathsf{tr}} (\mathcal{P}'; \Phi')$ such that $\Phi \sim_s \Phi'$ (and conversely)

Trace equivalence

Trace equivalence between configurations: $K \approx_t K'$.

For any execution trace $K \xrightarrow{\mathsf{tr}} (\mathcal{P}; \Phi)$ there exists an execution $K' \xrightarrow{\mathsf{tr}} (\mathcal{P}'; \Phi')$ such that $\Phi \sim_s \Phi'$ (and conversely)

Static equivalence between frames: $\Phi \sim_s \Phi'$.

Any test that holds in Φ also holds in Φ' (and conversely).

Example:

$\{\mathsf{w}_1 \mapsto k; \mathsf{w}_2 \mapsto \langle n, \mathsf{h}(n, k)\rangle\} \not\sim_s \{\mathsf{w}_1 \mapsto k; \mathsf{w}_2 \mapsto \langle n', \mathsf{h}(n', k')\rangle\}$

$\longrightarrow$ with the test $\mathsf{h}(\mathsf{proj}_1(w_2), w_1) \stackrel{?}{=} \mathsf{proj}_2(w_2)$.

Trace equivalence

Trace equivalence between configurations: $K \approx_t K'$.

For any execution trace $K \xrightarrow{\mathsf{tr}} (\mathcal{P}; \Phi)$ there exists an execution $K' \xrightarrow{\mathsf{tr}} (\mathcal{P}'; \Phi')$ such that $\Phi \sim_s \Phi'$ (and conversely)

Static equivalence between frames: $\Phi \sim_s \Phi'$.

Any test that holds in Φ also holds in Φ' (and conversely).

Example:

$\{\mathsf{w}_1 \mapsto k; \mathsf{w}_2 \mapsto \langle n, \mathsf{h}(n, k)\rangle\} \not\sim_s \{\mathsf{w}_1 \mapsto k; \mathsf{w}_2 \mapsto \langle n', \mathsf{h}(n', k')\rangle\}$

$\longrightarrow$ with the test $\mathsf{h}(\mathsf{proj}_1(w_2), w_1) \stackrel{?}{=} \mathsf{proj}_2(w_2)$.

$$\{\mathsf{w}_1 \mapsto \langle n_1, \mathsf{h}(n_1, k)\rangle;\ \mathsf{w}_2 \mapsto \langle n_2, \mathsf{h}(n_2, k)\rangle\}$$
$$\sim_s$$
$$\{\mathsf{w}_1 \mapsto \langle n_1', \mathsf{h}(n_1', k)\rangle;\ \mathsf{w}_2 \mapsto \langle n_2', \mathsf{h}(n_2', k')\rangle\}$$

Modelling unlinkability using trace equivalence

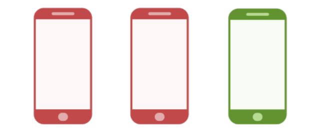
> Unlinkability aims to ensure *that a user may make multiple uses of a service or resource without others being able to link these uses together*. [ISO/IEC standard 15408]

$\longrightarrow$ the real system should be equivalent to the ideal one (from the point of view of the attacker).

Modelling unlinkability using trace equivalence

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]

$\longrightarrow$ the real system should be equivalent to the ideal one (from the point of view of the attacker).

 ... $\overset{?}{\approx}_t$  ...

many sessions
for each identity

only one session
for each identity

Modelling unlinkability (1^{th} attempt)

For single-step protocols, we may consider the following equivalence:

$$! \, \texttt{new} \, k; \; ! \; T(k) \;\approx_t\; ! \, \texttt{new} \, k; \; T(k)$$

$\longrightarrow$ This approach was used in [Chatzikokolakis *et al.*, 2010] to establish unlinkability for BH and OSK protocols.

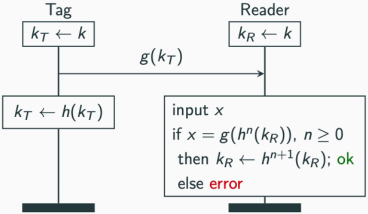
Modelling unlinkability (1^{th} attempt)

For single-step protocols, we may consider the following equivalence:

$$!\,\mathtt{new}\ k;\ !\ T(k)\ \approx_t\ !\,\mathtt{new}\ k;\ T(k)$$

$\longrightarrow$ This approach was used in [Chatzikokolakis *et al.*, 2010] to establish unlinkability for BH and OSK protocols.

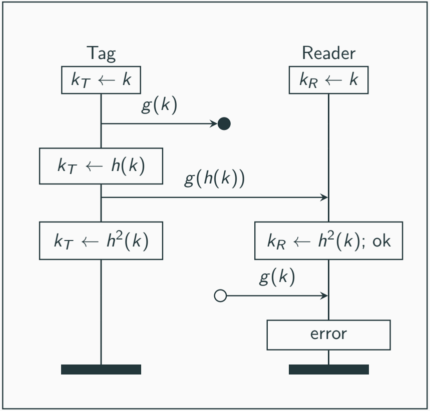
Example: OSK protocol



- h and g are two hash functions;
- k is updated - with $h(k)$ - after a successful execution on both sides.

Attack on the OSK protocol

Tags are proved unlinkable in [Chatzikokolakis *et al.*, 2010] but there is an attack !



Keypoint #1: modelling the reader is important.

Modelling unlinkability (2th attempt)

$\longrightarrow$ definition first proposed by [Arapinis *et al.*, CSF'10] (but for another notion of equivalence)

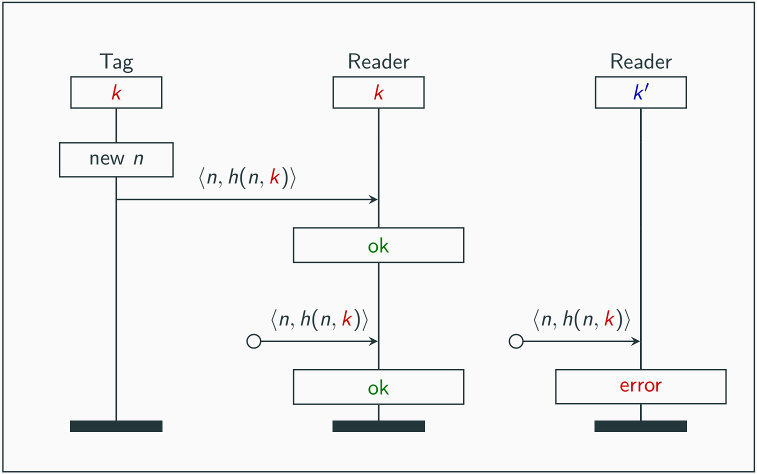
$$! \, \mathtt{new}\ k; \ (\, ! \, \mathsf{R}(k) \ | \ \ ! \, \mathsf{T}(k)) \ \approx_t \ ! \, \mathtt{new}\ k; \ (\, \mathsf{R}(k) \ | \ \mathsf{T}(k) \,)$$

This definition is:

- suitable to analyse e.g. e-passport protocols, and many other stateless protocols;
- the one we used in our work [Hirschi, Baelde & D., SP'16 & JCS'19].

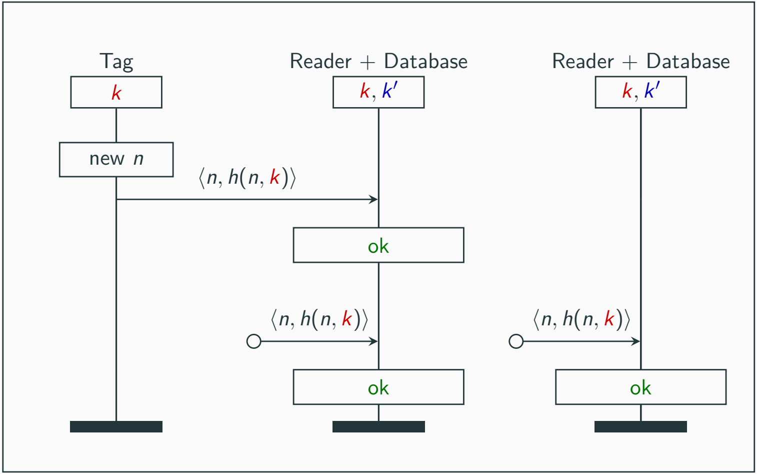
Going back to Basic Hash protocol (a stateful protocol)

$\longrightarrow$ linkable according to our previous definition (specific readers).



Basic Hash protocol

$\longrightarrow$ with a generic reader, no linkability attack.



Keypoint #2: The way the reader is modelled is important.

Modelling unlinkability for stateful protocols (3rd attempt)

$\longrightarrow$ definition proposed in [Baelde, D., Moreau, CSF'20]

We consider a generic reader having an access to a database DB

$$! \mathsf{R} \mid (\, !\, \mathtt{new}\ k; \mathtt{insert}\ \mathsf{DB}(k); !\, \mathsf{T}(k))$$
$$\approx_t$$
$$! \mathsf{R} \mid (\, !\, \mathtt{new}\ k; \mathtt{insert}\ \mathsf{DB}(k);\ \mathsf{T}(k))$$

Modelling unlinkability for stateful protocols (3rd attempt)

$\longrightarrow$ definition proposed in [Baelde, D., Moreau, CSF'20]

We consider a generic reader having an access to a database DB

$$!\,\mathsf{R} \mid (\,!\,\mathtt{new}\ k; \mathtt{insert}\ \mathsf{DB}(k); !\,\mathsf{T}(k))$$
$$\approx_t$$
$$!\,\mathsf{R} \mid (\,!\,\mathtt{new}\ k; \mathtt{insert}\ \mathsf{DB}(k);\ \mathsf{T}(k))$$

Basic Hash Example

- $\mathsf{R} = \mathtt{in}(c, y); \mathsf{get}\ z_k \in \mathsf{DB}\ \mathsf{suchthat}\ \mathsf{h}(\mathsf{proj}_1(y), z_k) = \mathsf{proj}_2(y)$
  $\mathsf{in}\ \mathtt{out}(c, \mathsf{ok})$
  $\mathsf{else}\ \mathtt{out}(c, \mathsf{ko})$.

$\longrightarrow$ Modelling tables in ProVerif (or Tamarin) is not an issue.

Part II

How can we establish unlinkability?

Exsiting tools able to establish trace equivalence

The problem is undecidable in general.

Exsiting tools able to establish trace equivalence

The problem is undecidable in general.

Approach 1: Limiting the number of sessions

- the problem becomes decidable (under some assumptions);
- decision procedures and tools have been developed, e.g. Deepsec, Spec, Sat-Equiv, ...

Exsiting tools able to establish trace equivalence

The problem is undecidable in general.

Approach 1: Limiting the number of sessions

- the problem becomes decidable (under some assumptions);
- decision procedures and tools have been developed, e.g. Deepsec, Spec, Sat-Equiv, . . .

Approach 2: Trying to solve the general case

- ProVerif: over-approximations are performed, termination is not guaranteed [Blanchet *et al.*, 2005]
- Tamarin: an interactive tool [Basin *et al.*, 2015]

$\longrightarrow$ they are based on diff-equivalence (too strong)

Diff-equivalence

How does it work (or not)?

- form a bi-process B using the operator $\mathsf{diff}[M_\mathsf{L}, M_\mathsf{R}]$;
- both sides of the bi-process B have to evolve simulatenously (+ static equivalence) to be declared in diff-equivalence

$\longrightarrow$ In such a case, we have that $\mathsf{fst}(B) \approx_t \mathsf{snd}(B)$.

Diff-equivalence

How does it work (or not)?

- form a bi-process B using the operator $\mathsf{diff}[M_{\mathsf{L}}, M_{\mathsf{R}}]$;
- both sides of the bi-process B have to evolve simulatenously (+ static equivalence) to be declared in diff-equivalence

$\longrightarrow$ In such a case, we have that $\mathsf{fst}(B) \approx_t \mathsf{snd}(B)$.

Formally, the semantics is given by a labelled transition system over bi-configurations $(\mathcal{P}; \Phi)$ where messages and computations may contain the diff operator.

Example 1: $\mathsf{out}(a) \mid \mathsf{out}(b) \stackrel{?}{\approx} \mathsf{out}(b) \mid \mathsf{out}(a)$

$\longrightarrow B = \mathsf{out}(\mathsf{diff}[a, b]) \mid \mathsf{out}(\mathsf{diff}[b, a])$ (* not in diff-equivalence *)

Diff-equivalence

How does it work (or not)?

- form a bi-process B using the operator $\mathsf{diff}[M_{\mathsf{L}}, M_{\mathsf{R}}]$;
- both sides of the bi-process B have to evolve simulatenously (+ static equivalence) to be declared in diff-equivalence

$\longrightarrow$ In such a case, we have that $\mathsf{fst}(B) \approx_t \mathsf{snd}(B)$.

Formally, the semantics is given by a labelled transition system over bi-configurations $(\mathcal{P}; \Phi)$ where messages and computations may contain the diff operator.

Example 2

$$B = \texttt{insert}\ DB(\mathsf{diff}[a, b]); \texttt{insert}\ DB(\mathsf{diff}[b, a]);$$
$$\text{get } x \text{ suchthat } x = a \text{ then } \mathsf{out}(c, ok) \text{ else } \mathsf{out}(c, ko)$$

(* not in diff-equivalence *)

Diff-equivalence does not hold on Basic Hash

$$B \; = !R \;|\; (!\,\texttt{new}\; k;\; !\texttt{new}\; kk; \texttt{insert}\; \mathrm{DB}(\mathrm{diff}[k, kk]); \mathrm{T}(\mathrm{diff}[k, kk])$$

Let's consider a scenario with:

- 1 reader;
- 2 tags: $T(\mathrm{diff}[k, kk_1])$, and $T(\mathrm{diff}[k, kk_2])$.

DB	left	right
line 1	k	kk_1
line 2	k	kk_2

Diff-equivalence does not hold on Basic Hash

$B = !R \mid (!\texttt{new}\ k; !\texttt{new}\ kk; \texttt{insert}\ \mathrm{DB}(\mathrm{diff}[k, kk]); \mathrm{T}(\mathrm{diff}[k, kk])$

Let's consider a scenario with:

- 1 reader;
- 2 tags: $T(\mathrm{diff}[k, kk_1])$, and $T(\mathrm{diff}[k, kk_2])$.

DB	left	right
line 1	k	kk_1
line 2	k	kk_2

1. The tag outputs $w_1 = \langle n_1, \mathrm{h}(n_1, \mathrm{diff}[k, kk_1])\rangle$;
2. The reader R will diverge on this input:

$\mathrm{R} = \texttt{in}(c, y);$
$\texttt{get}\ DB(z_k)\ \texttt{st.}\ \mathrm{eq}(\mathrm{h}(\mathrm{proj}_1(y), z_k), \mathrm{proj}_2(y))\ \texttt{in}\ \texttt{out}(c, \mathrm{ok})\ \texttt{else}\ \texttt{out}(c, \mathrm{ko})$

Diff-equivalence does not hold on Basic Hash

$$B \;= !R \;\mid\; (!\,\texttt{new}\; k;\; !\texttt{new}\; kk; \texttt{insert}\; \mathrm{DB}(\mathrm{diff}[k, kk]); \mathrm{T}(\mathrm{diff}[k, kk])$$

Let's consider a scenario with:

- 1 reader;
- 2 tags: $T(\mathrm{diff}[k, kk_1])$, and $T(\mathrm{diff}[k, kk_2])$.

DB	left	right
line 1	k	kk_1
line 2	k	kk_2

1. The tag outputs $w_1 = \langle n_1, \mathrm{h}(n_1, \mathrm{diff}[k, kk_1])\rangle$;
2. The reader R will diverge on this input:

$\mathrm{R} = \texttt{in}(c, y);$
get $DB(z_k)$ $\texttt{st.}$ $\mathrm{eq}(\mathrm{h}(\mathrm{proj}_1(y), z_k), \mathrm{proj}_2(y))$ $\texttt{in}$ $\mathrm{out}(c, \mathrm{ok})$ $\texttt{else}$ $\mathrm{out}(c, \mathrm{ko})$

$\longrightarrow$ Proverif returns `cannot be proved`.

Our result for stateless 2-party protocols

[Hirschi, Baelde, D.; S&P, 2016, JCS'19]

Theorem

If a protocol ensures both well-authentication and frame opacity then it ensures unlinkability, i.e.:

$$! \, \mathtt{new}\ k; \left(\,! \, \mathrm{R}(k) \;\mid\; ! \, \mathrm{T}(k) \right) \approx_t \; ! \, \mathtt{new}\ k; \left(\mathrm{R}(k) \;\mid\; \mathrm{T}(k) \right)$$

$\longrightarrow$ These 2 conditions are easier to check by existing tools

Intuition behind the sufficient conditions

Well-Authentication

- Goal = avoid leaks through outcomes of conditionals.
- "Whenever a conditional is positively evaluated, the agents involved are having so far an honest interaction."

$\longrightarrow$ This is a reachability property.

Intuition behind the sufficient conditions

Well-Authentication

- Goal = avoid leaks through outcomes of conditionals.
- "Whenever a conditional is positively evaluated, the agents involved are having so far an honest interaction."

$\longrightarrow$ This is a reachability property.

Frame Opacity

- Goal = avoid leaks through relations over messages.
- "Any reachable frame must be statically equivalent to an idealised frame that only depends on data already observed during the execution."

$\longrightarrow$ This can be verified with (an extension of) diff-equivalence.

Summary of our case studies using ProVerif

Protocol	WA	FO	unlinkability
Feldhofer	✓	✓	**safe**
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**

Our result for stateful 2-party protocols

[Baelde, D., Moreau, CSF'20]

Theorem

If a protocol ensures well-authentication, frame opacity and no desynchronisation then it ensures unlinkability, i.e.:

$$\begin{aligned} & !\,\mathsf{R} \mid (\,!\,\mathtt{new}\ k; \mathtt{insert}\ \mathsf{DB}(k); !\,\mathsf{T}(k)) \\ \approx_t\ & !\,\mathsf{R} \mid (\,!\,\mathtt{new}\ k; \mathtt{insert}\ \mathsf{DB}(k);\ \mathsf{T}(k)) \end{aligned}$$

No desynchronisation

- Goal = avoid leaks through desynchronisations between agents.
- "An honest interaction between a tag and a reader cannot fail."

$\longrightarrow$ This is also a reachability property! (But a little more tricky...)

Summary of our case studies using Tamarin

Protocol	WA	FO	ND	unlinkability
Basic Hash	✓	✓	✓	**safe**
Hash Lock	✓	✓	✓	**safe**
Feldhofer	✓	✓	✓	**safe**
OSK v1	✓		✗	attack
OSK v2	✓	✓	✓	**safe**
LAK (pairs)	✓		✗	attack
LAK (pairs, fixed)	✓	✓	✓	**safe**
LAK (pairs, no update)	✓	✓	✓	**safe**
5G-AKA (simplified)	✓	✓	✓	**safe**

$\longrightarrow$ simple conditions in the theory but not so easily checkable in practice

A recent result for stateful 2-party protocols

[Baelde, Debant, D., CSF'23]

Main Goal

Transform a ProVerif model $\mathcal{M}$ into another model $\mathcal{M}'$ such that:

- diff-equivalence on $\mathcal{M}' \Rightarrow$ trace equivalence on $\mathcal{M}$; and
- diff-equivalence is verified with ProVerif on $\mathcal{M}'$.

Our transformation:

1. duplicate the get instructions in $\mathcal{M}$ to dissociate the two parts of the bi-process; (possible using the `allowDiffPatterns` option)
2. add some axioms (proved correct manually) to help ProVerif to reason on our new model.

Basic Hash example

```
let T(k) = new nT; out(c,(nT,h(nT,k))).

let R = in(c,y);
 get db(k) suchthat snd(y) = h(fst(y),k) in out(c,ok)
             else out(c,ko).

process
   (! new k; ! new kk; insert db(diff[k,kk]);
           phase 1; T(diff[k,kk]))
        | (phase 1; ! R)
```

Step 1: Basic Hash example

$\longrightarrow$ duplicate the get instructions to dissociate the two parts of the bi-process.

```
let R =
  in(c,diff[y1L,y1R]);
  get db(diff[kL,wR]) suchthat snd(y1L) = h(fst(y1L),kL) in
    (get db(diff[wL,kR]) suchthat snd(y1R) = h(fst(y1R),kR) in
      out(c,diff[ok,ok])
    else
      out(c,diff[ok,ko]))
  else
    (get db(diff[wL,kR]) suchthat snd(y1R) = h(fst(y1R),kR) in
      out(c,diff[ko,ok])
    else
      out(cR,diff[ko,ko])).
```

Step 2: Refining the analysis in the failure branches

We illustrate this on a very simple example.

Before, ...

$$\begin{aligned} B = \ & \texttt{insert}\ \mathit{tbl}(\mathsf{ok}); \\ & \mathsf{get}\ \mathit{tbl}(x)\ \mathsf{st.\ true\ in}\ \texttt{out}(c, \mathsf{ok}) \\ & \qquad\qquad \mathsf{else}\ \texttt{out}(c, \mathsf{diff}[\mathsf{ok}_\mathsf{L}, \mathsf{ok}_\mathsf{R}]) \end{aligned}$$

... and ProVerif can not proved equivalence (whereas it holds).

Step 2: Refining the analysis in the failure branches

We illustrate this on a very simple example.

After, ...

$$B = \mathtt{event}(\mathsf{Inserted}(\mathsf{ok})); \ \mathtt{insert}\ tbl(\mathsf{ok});$$
$$\quad \mathsf{get}\ tbl(x)\ \mathsf{st.\ true\ in}\ \mathtt{out}(c, \mathsf{ok})$$
$$\qquad \mathsf{else}\ \mathtt{event}(\mathsf{Fail}()); \mathtt{out}(c, \mathsf{diff}[\mathsf{ok}_{\mathsf{L}}, \mathsf{ok}_{\mathsf{R}}])$$

... together with the following axiom:

$$\mathtt{event}(\mathsf{Fail}()) \wedge \mathtt{event}(\mathsf{Inserted}(\mathtt{diff}[y^{\mathsf{L}}, y^{\mathsf{R}}])) \Rightarrow \mathsf{false}.$$

$\longrightarrow$ Now, Proverif is able to conclude that equivalence holds.

Step 2: Refining the analysis in the failure branches

We illustrate this on a very simple example.

After, ...

$$B = \mathtt{event}(\mathsf{Inserted}(\mathsf{ok})); \mathtt{insert}\ tbl(\mathsf{ok});$$
$$\mathsf{get}\ tbl(x)\ \mathsf{st.\ true\ in}\ \mathtt{out}(c, \mathsf{ok})$$
$$\mathsf{else}\ \mathtt{event}(\mathsf{Fail}()); \mathtt{out}(c, \mathsf{diff}[\mathsf{ok_L}, \mathsf{ok_R}])$$

... together with the following axiom:

$$\mathtt{event}(\mathsf{Fail}()) \wedge \mathtt{event}(\mathsf{Inserted}(\mathtt{diff}[y^{\mathsf{L}}, y^{\mathsf{R}}])) \Rightarrow \mathsf{false}.$$

$\longrightarrow$ Now, Proverif is able to conclude that equivalence holds.

Going back to the Basic Hash protocol

$$\mathtt{event}(\mathsf{FailL}(x^{\mathsf{L}})) \wedge \mathtt{event}(\mathsf{Inserted}(\mathtt{diff}[y^{\mathsf{L}}, y^{\mathsf{R}}])) \Rightarrow \mathsf{proj}_2(x^{\mathsf{L}}) \neq \mathsf{h}(\mathsf{proj}_1(x^{\mathsf{L}}), y^{\mathsf{L}})$$
$$\mathtt{event}(\mathsf{FailR}(x^{\mathsf{R}})) \wedge \mathtt{event}(\mathsf{Inserted}(\mathtt{diff}[y^{\mathsf{L}}, y^{\mathsf{R}}])) \Rightarrow \mathsf{proj}_2(x^{\mathsf{R}}) \neq \mathsf{h}(\mathsf{proj}_1(x^{\mathsf{R}}), y^{\mathsf{R}})$$

Case studies

Implementation

The two steps of the transformation have been implemented ($\approx$ 2k Ocaml LoC).

Case studies

Basic Hash, Hash-Lock, Feldhofer, a variant of LAK, OSK.

$\longrightarrow$ ProVerif is able to conclude on all these examples !

Case studies

Implementation

The two steps of the transformation have been implemented ($\approx$ 2k Ocaml LoC).

Case studies

Basic Hash, Hash-Lock, Feldhofer, a variant of LAK, OSK.

$\longrightarrow$ ProVerif is able to conclude on all these examples !

(during a break if someone is interested)

Conclusion

Summary

- modelling unlinkability is rather subtle:
  $\longrightarrow$ importance of modelling the reader, and how it is modelled;
  $\longrightarrow$ states can introduce observables, especially in the case of a desynchronisation.
- verifying unlinkability properties is not an easy task but a lot of progress has been done.

Summary

- modelling unlinkability is rather subtle:
  $\longrightarrow$ importance of modelling the reader, and how it is modelled;
  $\longrightarrow$ states can introduce observables, especially in the case of a desynchronisation.
- verifying unlinkability properties is not an easy task but a lot of progress has been done.

Going a step further:

- stateful protocols (with updates) using ProVerif/GSVerif;
- from diff-equivalence to session equivalence;
- A nice way to encore unlinkability in Tamarin is to rely on (asymmetric) restrictions but currently the tool does not support them.

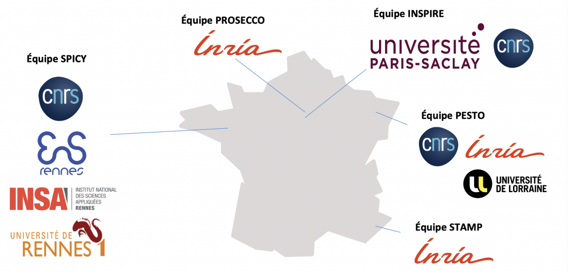
Advertisement

PEPR Cybersecurity (2022-2028)

Partners: 5 teams in France (Nancy, Paris, Rennes, Sophia) `https://pepr-cyber-svp.cnrs.fr`

Job offers:

- PhDs
- Post-docs
- Engineers



⟶ contact me: stephanie.delaune@irisa.fr