Establishing unlinkability relying on existing verification tools

Stéphanie **DELAUNE**

Univ Rennes, CNRS, IRISA, France

$\longrightarrow$ joint work with D. Baelde, 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,
  $\hookrightarrow$ 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

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 !**

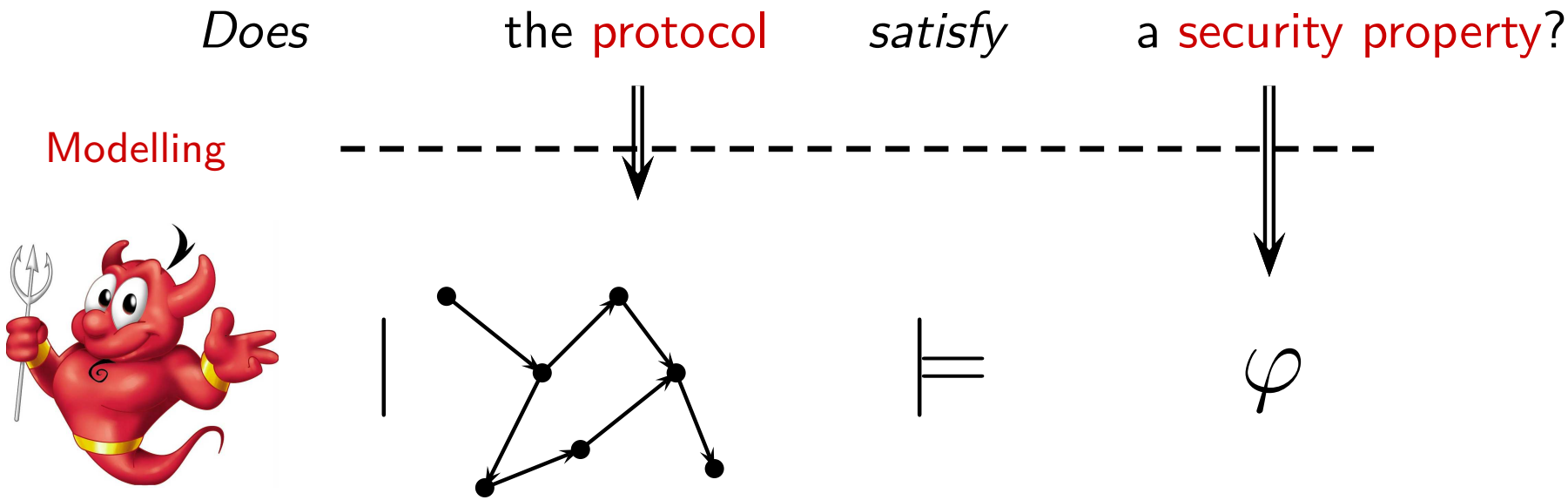
- perform a manual security analysis
  $\longrightarrow$ **this is error-prone !**

Our approach

formal symbolic verification using automatic tools.

$\longrightarrow$ provides a rigorous framework and automatic tools to analyse security protocols and find their logical flaws.

Formal symbolic verification in a nutshell



Two main tasks:

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

Some success stories

Some success stories

TAMARIN
Tamarin prover interactive mode

ProVerif

AVANTSSAR

Authentication flaw in the Single Sign-On protocol used *e.g.* in GMail [Armando *et al.*, 2011]

A formal analysis of 5G Authentication (using Tamarin) [Basin *et al.*, 2018]

Project miTLS – a verified reference implementation of the TLS protocol

`https://www.mitls.org/`

What about 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]

the real system should be indistinguishable from the ideal one – from the point of view of the attacker.

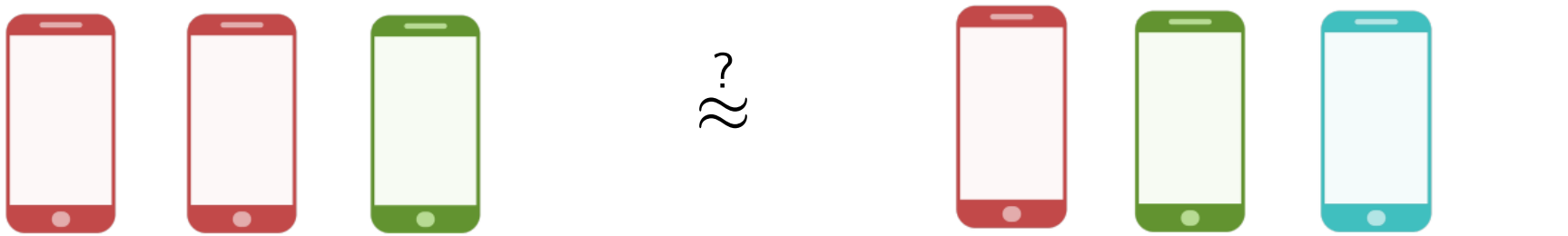
$\longrightarrow$ usually expressed using a notion of equivalence

What about 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]

the real system should be indistinguishable from the ideal one – from the point of view of the attacker.

$\longrightarrow$ usually expressed using a notion of equivalence



many sessions for each identity

only one session for each identity

A challenging problem . . .

. . . which is undecidable in general

A challenging problem . . .

. . . which is undecidable in general

Approach 1: Limiting the number of sessions

- the problem becomes decidable (under some restrictions);
- decision procedures and tools have been developed, e.g. Deepsec, Akiss, . . .

$\longrightarrow$ existing tools scale badly

A challenging problem . . .

. . . which is undecidable in general

Approach 1: Limiting the number of sessions

- the problem becomes decidable (under some restrictions);
- decision procedures and tools have been developed, e.g. Deepsec, Akiss, . . .

$\longrightarrow$ existing tools scale badly

Approach 2: Trying to solve the general case

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

$\longrightarrow$ they are based on diff-equivalence – too strong in practice

A challenging problem . . .

. . . which is undecidable in general

Approach 1: Limiting the number of sessions

- the problem becomes decidable (under some restrictions);
- decision procedures and tools have been developed, e.g. Deepsec, Akiss, . . .

$\longrightarrow$ existing tools scale badly

Approach 2: Trying to solve the general case

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

$\longrightarrow$ they are based on diff-equivalence – too strong in practice

None of these tools are suitable to analyse unlinkability.

Some related works about unlinkability

Actually, few formal proofs exist in the unbounded setting.

- [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
- [Arapinis *et al.*, 2012]: a formal analysis of a simplified version of the AKA protocol using the diff-equivalence of ProVerif.

Some related works about unlinkability

Actually, few formal proofs exist in the unbounded setting.

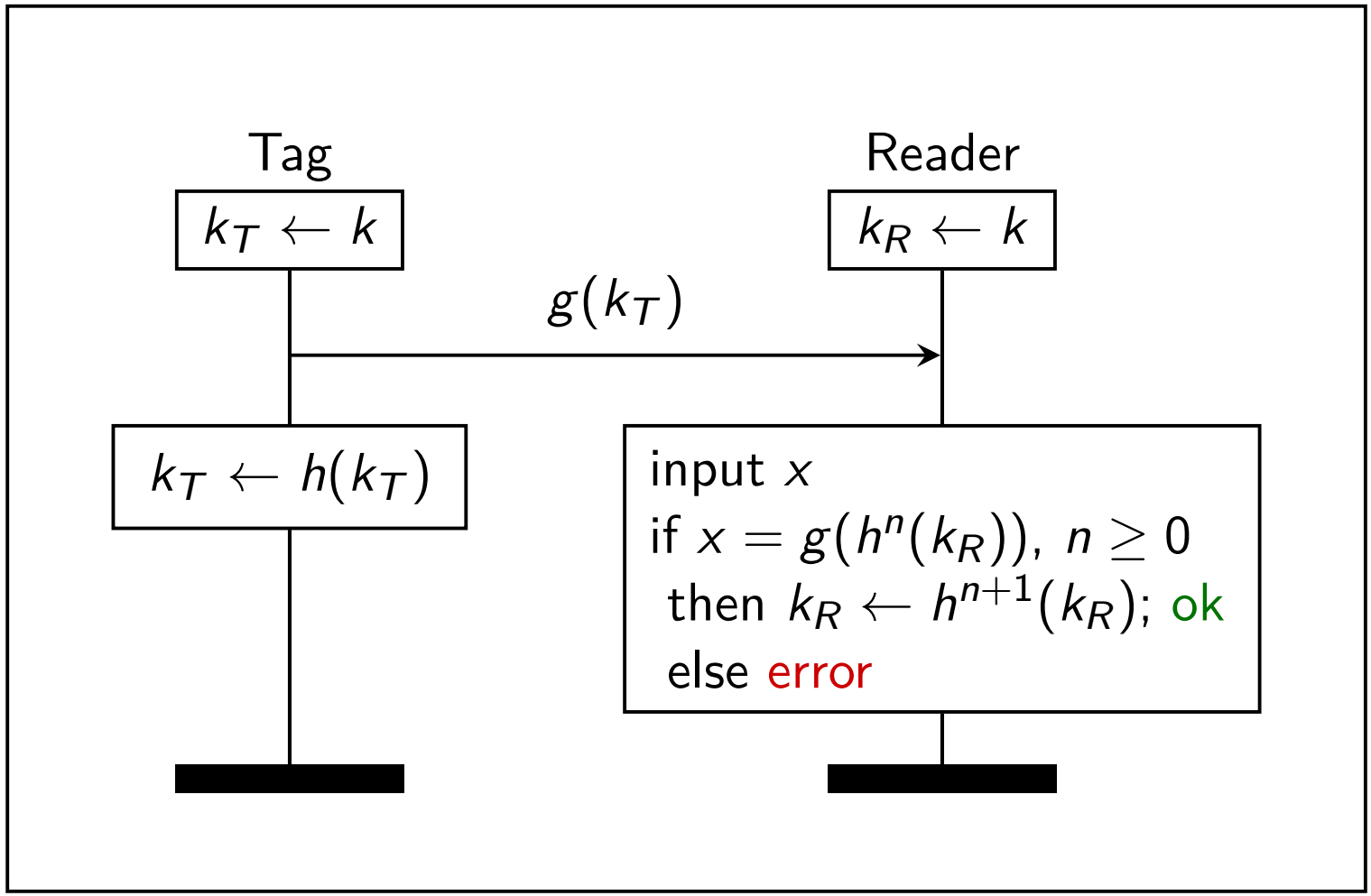
- [Arapinis *et al.*, 2010]: a formal definition of unlinkability, and a manual proof of unlinkability for a fixed version of the e-passport protocol. $\longrightarrow$ this result is wrong
- [Arapinis *et al.*, 2012]: a formal analysis of a simplified version of the AKA protocol using the diff-equivalence of ProVerif.
- [Koutsos, 2017 & 2018]: manual and very detailed proofs of unlinkability for two RFID protocols, and a fixed version of AKA.

Some related works about unlinkability

Actually, few formal proofs exist in the unbounded setting.

- [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
- [Arapinis *et al.*, 2012]: a formal analysis of a simplified version of the AKA protocol using the diff-equivalence of ProVerif.
- [Koutsos, 2017 & 2018]: manual and very detailed proofs of unlinkability for two RFID protocols, and a fixed version of AKA.
- [Chatzikokolakis *et al.*, 2010]: sufficient conditions checkable using ProVerif that allows us to establish unlinkability for a simple class of protocols (single-step protocols).
- ...

OSK protocol [Ohkubo *et al.* 2003]



$\longrightarrow$ h and g are two hash functions

Our contributions

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

Our contributions

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

On the theoretical side

some reasonable conditions implying 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. An extension of this result to deal with stateful protocols is currently under submission.

Part I

Modelling: protocols, security properties, and the attacker

Protocols as processes

P, Q	$:=$	0	null process
	$\mid$	$\mathsf{new}\, n.P$	fresh name
	$\mid$	$\mathsf{in}(c, x).P$	input
	$\mid$	$\mathsf{out}(c, u).P$	output
	$\mid$	$\mathsf{let}\, \overline{x} = \overline{t}\ \mathsf{in}\ P\ \mathtt{else}\ Q$	evaluation
	$\mid$	$\mathsf{get}(r, x).P \mid \mathsf{set}(r, v).P$	memory cell
	$\mid$	$\mathsf{lookup} \ldots \mathsf{such\ that} \ldots \mathsf{in}\ P\ \mathtt{else}\ Q$	DB test and update
	$\mid$	$\mathsf{insert}(v).P$	DB insertion
	$\mid$	$(P \mid Q)$	parallel
	$\mid$	$!P$	replication
	$\mid$	$P; Q$	sequence
	$\mid$	$\mathsf{i}P$	repetition

Messages that are exchanged are modelled using terms.

Messages as terms

They are built from names $\mathcal{N}$, and elements in $\Sigma = \Sigma_c \uplus \Sigma_d$.

To reflect the properties of the cryptographic primitives:

- destructor symbols are subject to a computation relation $\Downarrow$:
- constructor terms are subject to an equational theory $=_E$:

Messages as terms

They are built from names $\mathcal{N}$, and elements in $\Sigma = \Sigma_c \uplus \Sigma_d$.

To reflect the properties of the cryptographic primitives:

- destructor symbols are subject to a computation relation $\Downarrow$:
- constructor terms are subject to an equational theory $=_E$:

Example:

$\Sigma_c = \{\mathsf{h}, \mathsf{g}, \mathsf{senc}, \mathsf{mac}, \langle\,\rangle, \oplus, 0, \mathsf{ok}\}$ and $\Sigma_d = \{\mathsf{sdec}, \mathsf{proj}_1, \mathsf{proj}_2, \mathsf{eq}\}$.

$$\begin{array}{rcl@{\qquad}rcl} \mathsf{sdec}(\mathsf{senc}(x,y),y) & \to & x & \mathsf{proj}_1(\langle x_1,x_2\rangle) & \to & x_1 \\ \mathsf{eq}(x,x) & \to & \mathsf{ok} & \mathsf{proj}_1(\langle x_1,x_2\rangle) & \to & x_2 \end{array}$$

$$\begin{array}{rcl@{\qquad}rcl} x \oplus (y \oplus z) & = & (x \oplus y) \oplus z & x \oplus 0 & = & x \\ x \oplus y & = & y \oplus x & x \oplus x & = & 0 \end{array}$$

Messages as terms

They are built from names $\mathcal{N}$, and elements in $\Sigma = \Sigma_c \uplus \Sigma_d$.

To reflect the properties of the cryptographic primitives:

- destructor symbols are subject to a computation relation $\Downarrow$:
- constructor terms are subject to an equational theory $=_E$:

Example:

$\Sigma_c = \{\mathsf{h}, \mathsf{g}, \mathsf{senc}, \mathsf{mac}, \langle\,\rangle, \oplus, 0, \mathsf{ok}\}$ and $\Sigma_d = \{\mathsf{sdec}, \mathsf{proj}_1, \mathsf{proj}_2, \mathsf{eq}\}$.

$$\begin{array}{rcl@{\qquad}rcl} \mathsf{sdec}(\mathsf{senc}(x,y),y) & \to & x & \mathsf{proj}_1(\langle x_1,x_2\rangle) & \to & x_1 \\ \mathsf{eq}(x,x) & \to & \mathsf{ok} & \mathsf{proj}_1(\langle x_1,x_2\rangle) & \to & x_2 \end{array}$$

$$\begin{array}{rcl@{\qquad}rcl} x \oplus (y \oplus z) & = & (x \oplus y) \oplus z & x \oplus 0 & = & x \\ x \oplus y & = & y \oplus x & x \oplus x & = & 0 \end{array}$$

$\mathsf{sdec}(\mathsf{senc}(s, k_1 \oplus k_2), k_2 \oplus k_1) \Downarrow s$ $\qquad$ $\mathsf{eq}(k_1 \oplus k_2, k_2 \oplus k_1) \Downarrow \mathsf{ok}$

Going back to the OSK protocol

A **tag** with a memory cell r:

$$\mathsf{Tag} \quad = \quad \mathsf{get}(r, y).\mathsf{set}(r, \mathsf{h}(y)).\mathsf{out}(c_T, \mathsf{g}(y))$$

Going back to the OSK protocol

A **tag** with a memory cell r:

$$\mathsf{Tag} \quad = \quad \mathsf{get}(r, y).\mathsf{set}(r, \mathsf{h}(y)).\mathsf{out}(c_T, \mathsf{g}(y))$$

A **reader** with a database DB:

$$\begin{aligned}\mathsf{Reader} \quad = \quad & \mathsf{in}(c_T, x).\text{lookup } y \text{ such that} \\ & x_{\mathsf{test}} = \mathsf{TestOSK}(y, x),\ y' = \mathsf{UpdateOSK}(x) \text{ in} \\ & \mathsf{out}(c_R, \mathsf{ok})\,, \mathtt{else}\, \mathsf{out}(c_R, \mathsf{error})\end{aligned}$$

with the following infinite set of rewriting rules:

- $\mathsf{TestOSK}(z, \mathsf{g}(\mathsf{h}^n(z))) \to \mathsf{ok}$ with $n \geq 0$; and
- $\mathsf{UpdateOSK}(\mathsf{g}(z)) \to \mathsf{h}(z)$.

Semantics – How processes can evolve?

$\longrightarrow$ a labelled transition system over configurations $K = (\mathcal{P}; \phi; \mathcal{S})$:

- $\mathcal{P}$ is a multiset of processes;
- $\phi = \{w_1 \mapsto u_1, \ldots, w_n \mapsto u_n\}$ is a frame;
- $\mathcal{S}$ is a store.

Semantics – How processes can evolve?

$\longrightarrow$ a labelled transition system over configurations $K = (\mathcal{P}; \phi; \mathcal{S})$:

- $\mathcal{P}$ is a multiset of processes;
- $\phi = \{w_1 \mapsto u_1, \ldots, w_n \mapsto u_n\}$ is a frame;
- $\mathcal{S}$ is a store.

$$\textsc{In} \qquad (\mathsf{in}(c, x).P \cup \mathcal{P}; \phi; \mathcal{S}) \xrightarrow{\mathsf{in}(c,R)} (P\{x \mapsto R\phi\Downarrow\} \cup \mathcal{P}; \phi; \mathcal{S})$$

$$\textsc{Out} \qquad (\mathsf{out}(c, u).P \cup \mathcal{P}; \phi; \mathcal{S}) \xrightarrow{\mathsf{out}(c,w)} (P \cup \mathcal{P}; \phi \cup \{w \mapsto u\}; \mathcal{S})$$

Trace equivalence

Static equivalence between frames – $\phi \sim \phi'$

A **test holds** in ϕ if, and only, if it holds in ϕ'.

Trace equivalence

Static equivalence between frames – $\phi \sim \phi'$

A **test holds** in ϕ if, and only, if it holds in ϕ'.

Example

$$\{w_1 \mapsto \mathsf{senc}(\mathsf{ok}, k), w_2 \mapsto k\} \stackrel{?}{\sim} \{w_1 \mapsto \mathsf{senc}(n, k'), w_2 \mapsto k'\}$$

Trace equivalence

Static equivalence between frames – $\phi \sim \phi'$

A **test holds** in ϕ if, and only, if it holds in ϕ'.

Example

$$\{w_1 \mapsto \mathsf{senc}(\mathsf{ok}, k), w_2 \mapsto k\} \not\sim \{w_1 \mapsto \mathsf{senc}(n, k'), w_2 \mapsto k'\}$$

$\longrightarrow$ with the test $\mathsf{sdec}(w_1, w_2) \stackrel{?}{=} \mathsf{ok}$.

Trace equivalence

Static equivalence between frames – $\phi \sim \phi'$

A **test holds** in ϕ if, and only, if it holds in ϕ'.

Example

$$\{w_1 \mapsto \mathsf{senc}(\mathsf{ok}, k), w_2 \mapsto k\} \not\sim \{w_1 \mapsto \mathsf{senc}(n, k'), w_2 \mapsto k'\}$$

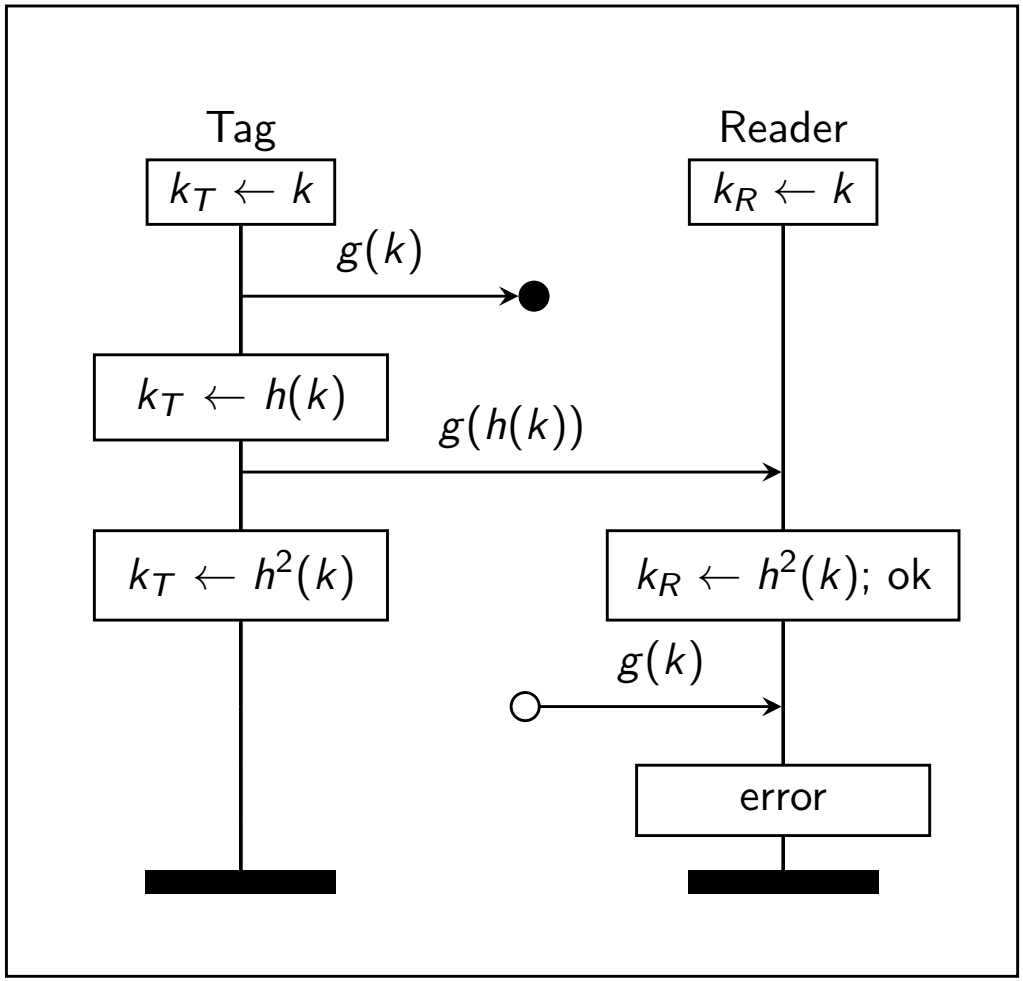
$\longrightarrow$ with the test $\mathsf{sdec}(w_1, w_2) \stackrel{?}{=} \mathsf{ok}$.

Trace equivalence between configurations – $K \approx K'$

For any execution $K \xrightarrow{\mathsf{tr}} (\mathcal{P}; \phi; \mathcal{S})$, there exists an execution $K' \xrightarrow{\mathsf{tr}} (\mathcal{P}'; \phi'; \mathcal{S}')$ such that $\phi \sim \phi'$.

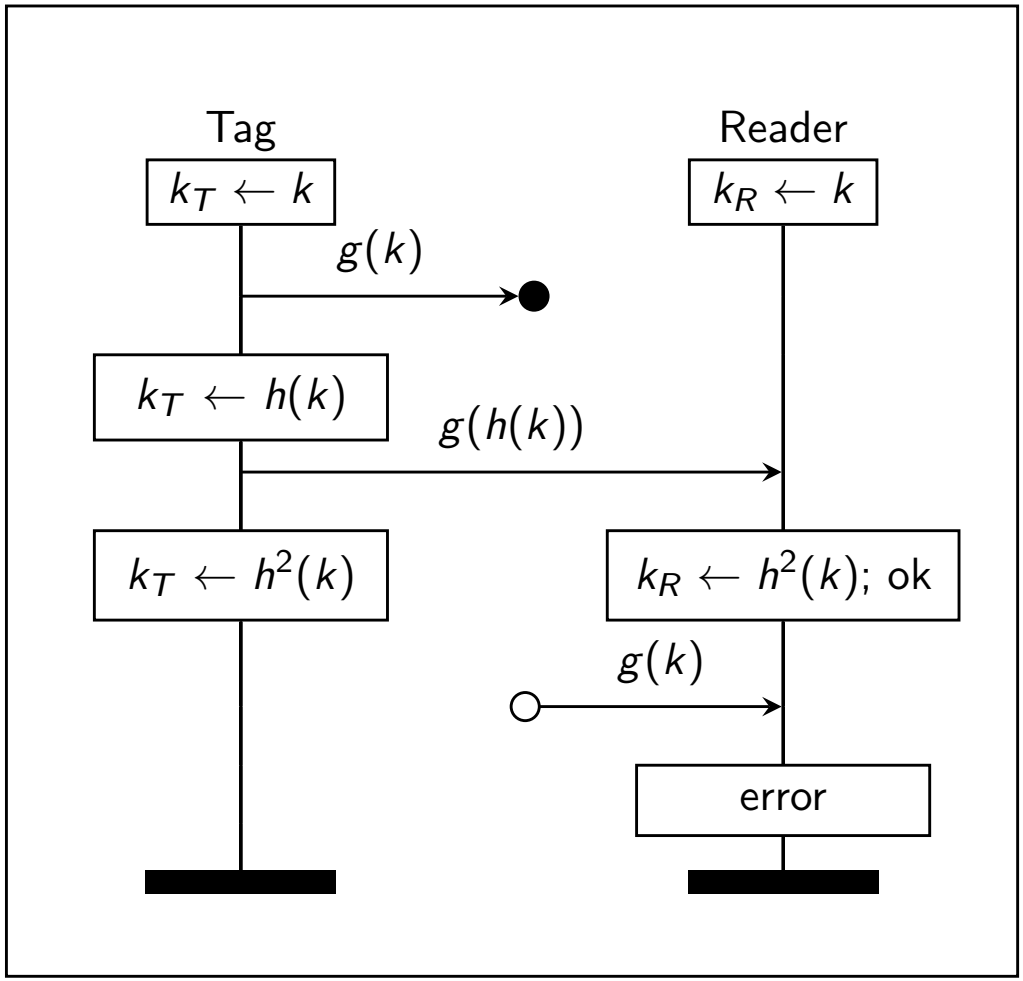
Going back to the OSK protocol

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



Going back to 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 for stateless protocols

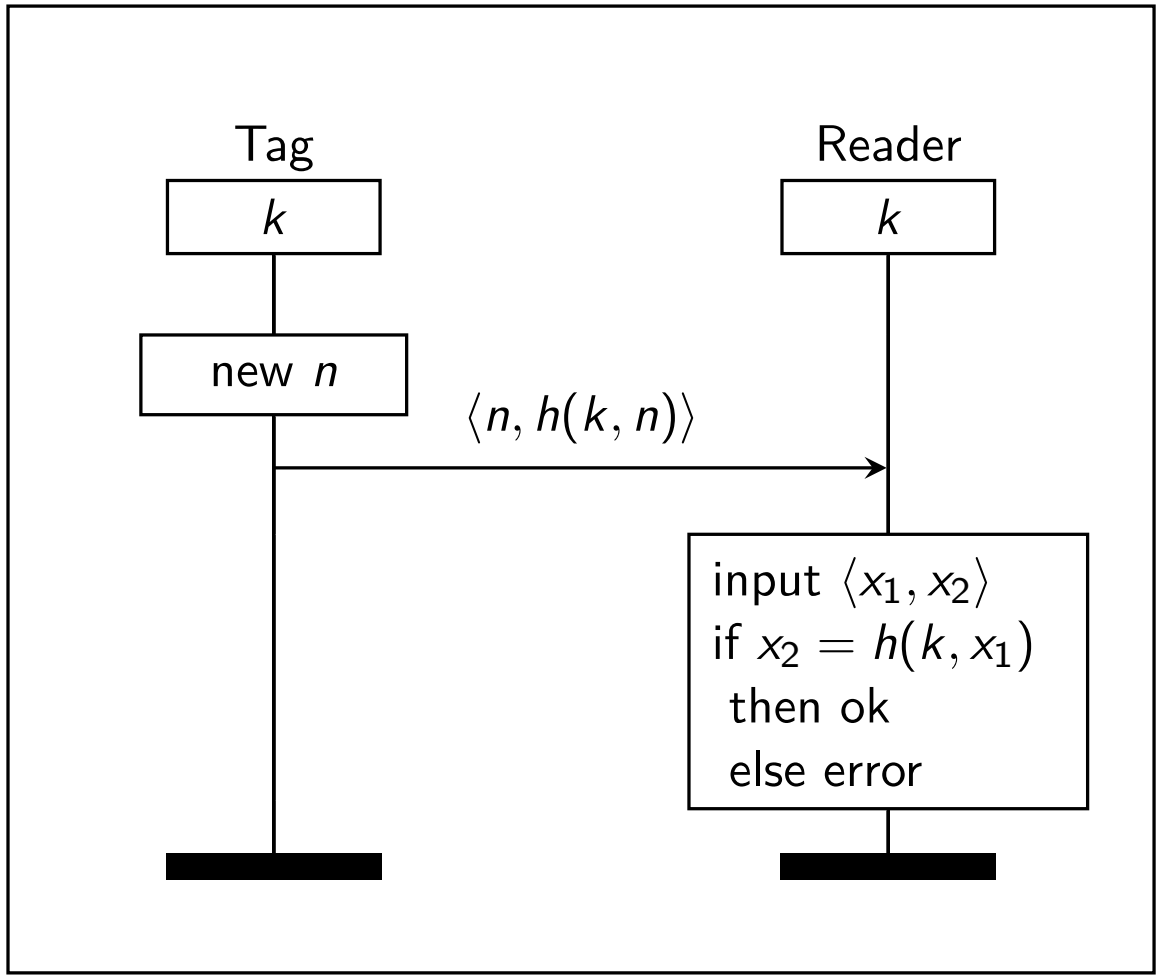
many sessions for each identity $\approx$ only one session for each identity

$$!\text{new}\,\overline{k}.(!\text{new}\,\overline{n}_T.\text{Tag} \mid !\text{new}\,\overline{n}_R.\text{Reader})$$
$$\approx$$
$$!\text{new}\,k.(\text{new}\,\overline{n}_T.\text{Tag} \mid \text{new}\,\overline{n}_R.\text{Reader})$$

$\longrightarrow$ this is the definition we used in our S&P paper to analyse e.g. e-passport protocols.

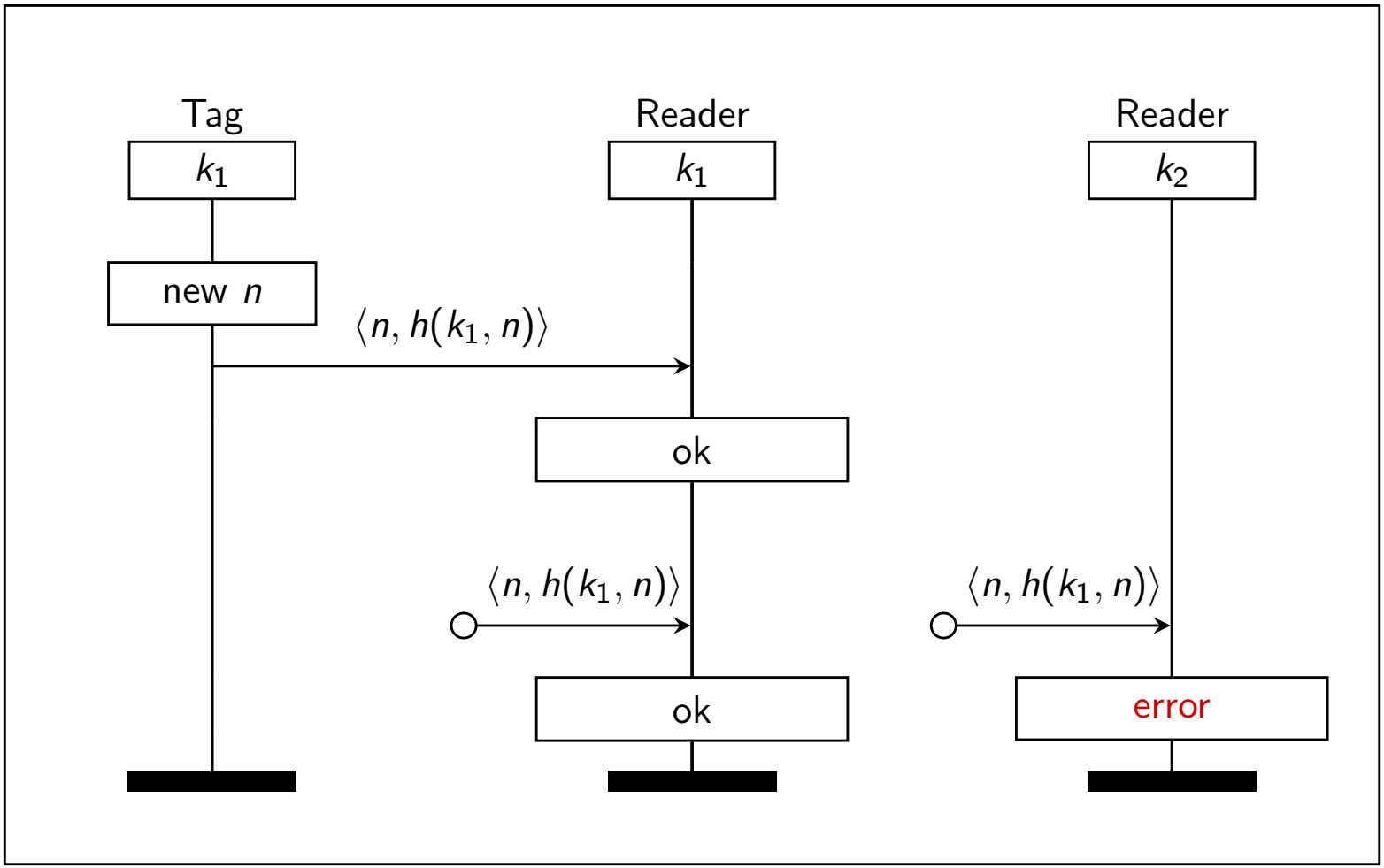
Basic Hash protocol (a stateful protocol)

$\longrightarrow$ but the state is never updated.



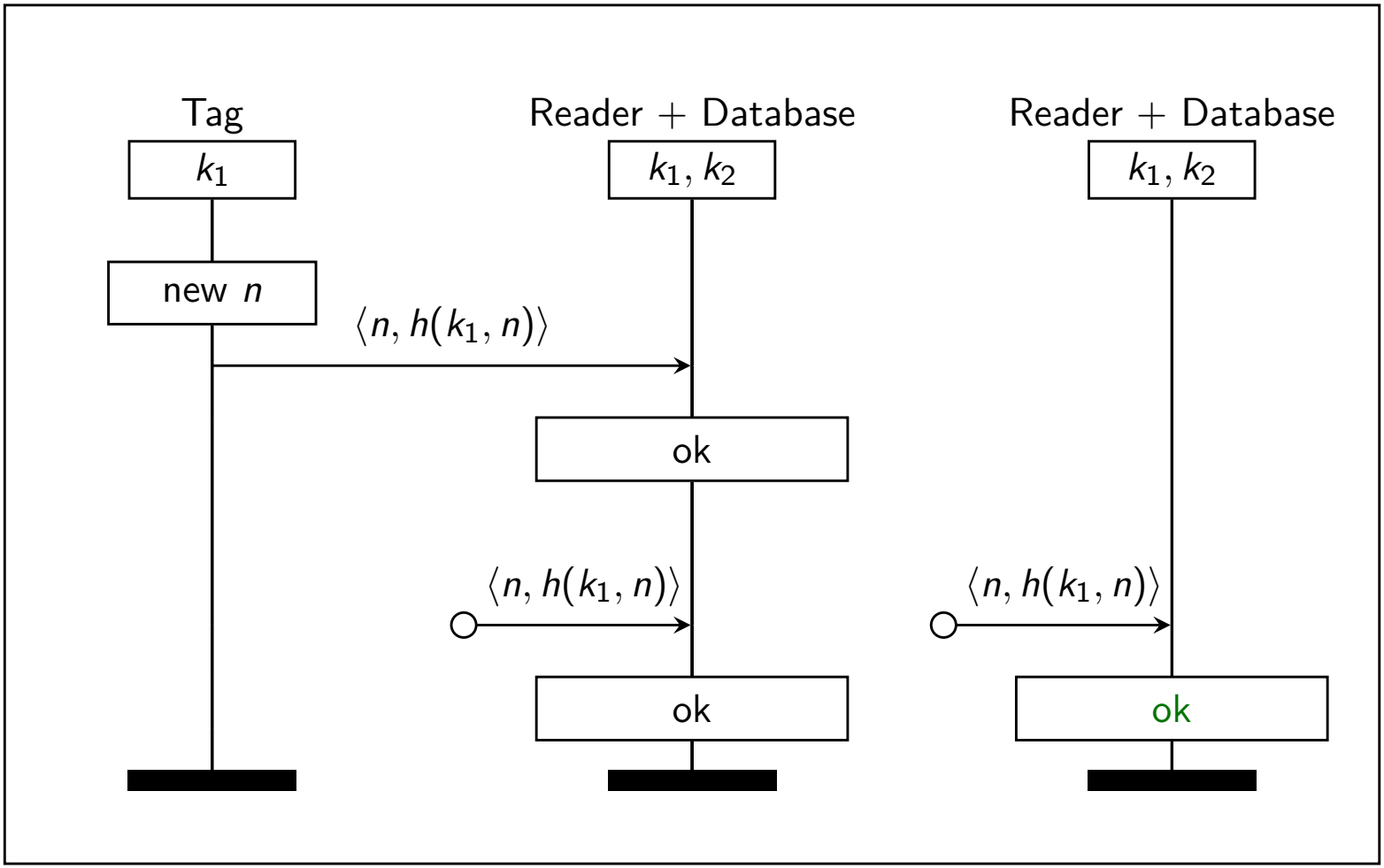
Basic Hash protocol

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



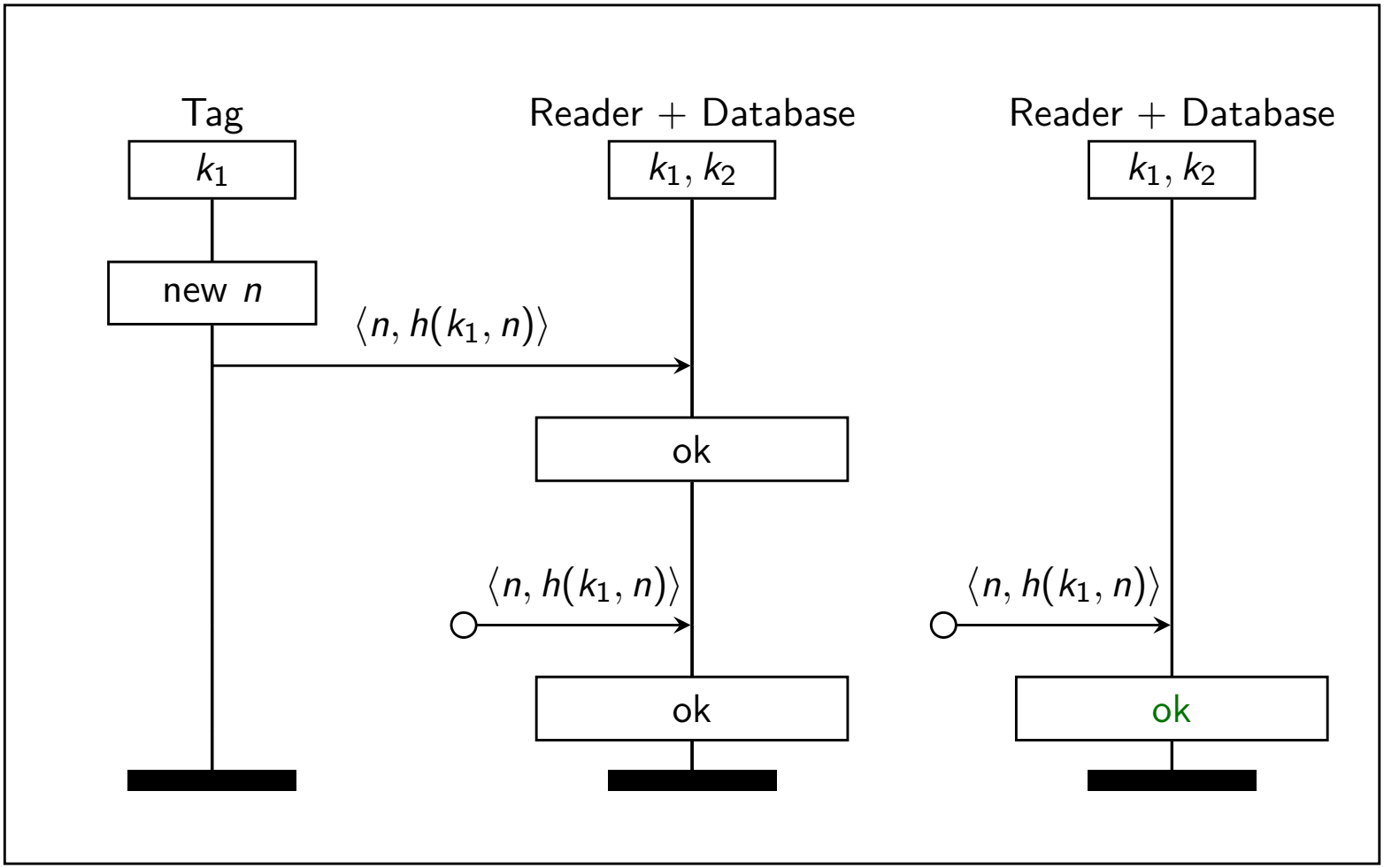
Basic Hash protocol

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



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

$\longrightarrow$ our previous definition is too strong, and thus we propose the following one:

A protocol Π ensures unlinkability if $\mathcal{M}_\Pi \approx \mathcal{S}_\Pi$.

$$\begin{aligned}\mathcal{M}_\Pi := \quad & (!\ \text{new}\,\overline{k}.\text{new}\,r.\text{set}(r, v_T).\text{insert}(v_R).\ \text{i}\ \text{new}\,\overline{n}_T.\text{Tag}) \\ & |\ (!\ \text{new}\,\overline{n}_R.\text{Reader})\end{aligned}$$

$$\begin{aligned}\mathcal{S}_\Pi := \quad & (!\ \text{new}\overline{k}.\text{new}\,r.\text{set}(r, v_T).\text{insert}(v_R).\quad \text{new}\,\overline{n}_T.\text{Tag}) \\ & |\ (!\ \text{new}\,\overline{n}_R.\text{Reader})\end{aligned}$$

Part II

How can we check trace equivalence?

for an unbounded number of sessions

Going back to diff-equivalence

How it works (or not)?

- form a bi-process B using the operator choice$[u_{left}, u_{right}]$;
- if B is in diff-equivalence then $B_{left} \approx B_{right}$

Going back to diff-equivalence

How it works (or not)?

- form a bi-process B using the operator choice$[u_{left}, u_{right}]$;
- if B is in diff-equivalence then $B_{left} \approx B_{right}$

Example - Basic Hash protocol

$$
\begin{array}{rcl}
T & = & \text{new } n.\text{out}(c, \langle n, \text{h}(k, n)\rangle) \\
R & = & \text{in}(c, x). \\
 & & \text{lookup } k \text{ such that } \text{proj}_2(x) = \text{h}(k, \text{proj}_1(x)) \text{ in } \text{out}(c, \text{ok})
\end{array}
$$

Going back to diff-equivalence

How it works (or not)?

- form a bi-process B using the operator choice$[u_{left}, u_{right}]$;
- if B is in diff-equivalence then $B_{left} \approx B_{right}$

Example - Basic Hash protocol

$$
\begin{array}{rcl}
T & = & \mathsf{new}\, n.\mathsf{out}(c, \langle n, \mathsf{h}(k, n)\rangle) \\
R & = & \mathsf{in}(c, x). \\
 & & \mathsf{lookup}\ k\ \mathsf{such\ that}\ \mathsf{proj}_2(x) = \mathsf{h}(k, \mathsf{proj}_1(x))\ \mathsf{in}\ \mathsf{out}(c, \mathsf{ok})
\end{array}
$$

$$
B = (!\ R) \mid (!\ \mathsf{new}k\ !\ \mathsf{new}kk.\mathsf{insert}(\mathsf{choice}[k, kk]).T(\mathsf{choice}[k, kk]))
$$

$$
\begin{array}{rcll}
B_{left} & = & (!\ R) \mid (!\ \mathsf{new}\, k\ !\ \mathsf{insert}(k).T(k)) & \longrightarrow \mathcal{M}_\Pi \\
B_{right} & = & (!\ R) \mid (!\ !\ \mathsf{new}\, kk.\mathsf{insert}(kk).T(kk)) & \longrightarrow \mathcal{S}_\Pi
\end{array}
$$

Why diff-equivalence is too strong?

$$B \quad = \quad (!\ R) \mid (!\ \text{new}\ k\ !\ \text{new}\ kk.\text{insert}(\text{choice}[k, kk]).T(\text{choice}[k, kk]))$$

Let's consider a scenario with:

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

Why diff-equivalence is too strong?

$$B \quad = \quad (!\ R) \mid (!\ \text{new}\ k\ !\ \text{new}\ kk.\text{insert}(\text{choice}[k, kk]).T(\text{choice}[k, kk]))$$

Let's consider a scenario with:

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

DB	left	right
line 1	k	kk_1
line 2	k	kk_2

The frame contains:

- $w_1 = \langle n_1, \text{h}((\text{choice}[k, kk_1], n_1))\rangle$.

Our result for stateless 2-party protocols

Definition

$\mathcal{M}_\Pi := !\ \text{new}\, \overline{k}.(!\ \text{new}\, \overline{n}_T.\text{Tag} \mid !\ \text{new}\, \overline{n}_R.\text{Reader})$

$\mathcal{S}_\Pi := !\ \text{new}\, \overline{k}.(\text{new}\, \overline{n}_T.\text{Tag} \mid \ \text{new}\, \overline{n}_R.\text{Reader})$

A protocol Π ensures **unlinkability** if $\mathcal{M}_\Pi \approx \mathcal{S}_\Pi$.

Theorem

If a protocol ensures both well-authentication and frame opacity then it ensures unlinkability.

$\longrightarrow$ 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."

$\Rightarrow$ 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."

$\Rightarrow$ 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."

$\Rightarrow$ 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

Definition

$$\mathcal{M}_\Pi := (!\ \text{new}\,\overline{k}.\text{new}\,r.\text{set}(r, v_T).\text{insert}(v_R).\ \text{i}\ \text{new}\,\overline{n}_T.\text{Tag}) \mid (!\ \text{new}\overline{n}_R.\text{Reader})$$

$$\mathcal{S}_\Pi := (!\ \text{new}\overline{k}.\text{new}\,r.\text{set}(r, v_T).\text{insert}(v_R).\quad \text{new}\,\overline{n}_T.\text{Tag}) \mid (!\ \text{new}\,\overline{n}_R.\text{Reader})$$

A protocol Π ensures **unlinkability** if $\mathcal{M}_\Pi \approx \mathcal{S}_\Pi$.

Theorem

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

Intuition behind no desynchronisation

No desynchronisation

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

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

Intuition behind no desynchronisation

No desynchronisation

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

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

Example: OSK protocol

There exists an execution where an honest interaction goes into the else branch (because the tag and the reader are desynchronised).

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**

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.
- a methodology to establish unlinkability: sufficient condition also simpler to verify by existing tools.
- validate by a number of case studies.

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.
- a methodology to establish unlinkability: sufficient condition also simpler to verify by existing tools.
- validate by a number of case studies.

Going a step further:

- Simple conditions in the theory but not so easily checkable in practice (especially when using Tamarin)
- Existing tools are not designed to verify our conditions.

Thank you for your attention!

Any questions?