A Method for Verifying Privacy-Type Properties: The Unbounded Case

Published at Security&Privacy’16

Lucca Hirschi, David Baelde and Stéphanie Delaune

8th December, 2016
we need formal verification of crypto protocols covering privacy
we need formal verification of crypto protocols covering privacy

Goal:

- checking **unlinkability** and **anonymity**
- in the **symbolic model** (≡ Dolev-Yao model)
- for **unbounded sessions and users**
we need formal verification of crypto protocols covering privacy

Goal:

- checking **unlinkability** and **anonymity**
- in the **symbolic model** (= Dolev-Yao model)
- for **unbounded sessions and users**

- **Unlinkability** (=untraceability) [ISO/IEC 15408]:
  
  Ensuring that a user may make multiple uses of a service or resource without others being able to link these uses together.

- **Anonymity** [ISO/IEC 15408]:
  
  Ensuring that a user may use a service or resource without disclosing the user’s identity. [..]
Big Picture

Protocol’s specification

Security goal
(e.g., Secrecy)
Big Picture

Protocol's specification

Protocol's model

(e.g., ProVerif model)

Reachability
in a model

Security goal

Modelization of the property

Reachability(State_{bad})
Big Picture

Protocol's specification $\rightarrow$ Protocol's model (e.g., ProVerif model)

Reachability in a model $\rightarrow$ Automatic Verification

Secure

No

Attack

Reachability?

Security goal (e.g., Secrecy) $\rightarrow$ Modelization of the property Reachability($\text{State}_{\text{bad}}$)
Big Picture

Privacy goal
(e.g., Unlinkability)

Modelization of the property
\( \nu id \nu Sess. P \approx \nu id. \nu Sess. P \)

Protocol's specification
Protocol's model
(e.g., ProVerif model)

Equivalence
of two models
Lucca Hirschi
A Method for Verifying Privacy-Type Properties: The Unbounded Case

Big Picture

Protocol’s specification ➔ Protocol’s model
(e.g., ProVerif model)

Privacy goal ➔ Modelization of the property
(e.g., Unlinkability)

Equivalence of two models

Automatic Verification

Secure

Protocol’s specification ➔ Protocol’s model
(e.g., ProVerif model)

Privacy goal ➔ Modelization of the property
(e.g., Unlinkability)

Equivalence of two models

Automatic Verification

Secure

Protocol’s specification ➔ Protocol’s model
(e.g., ProVerif model)

Privacy goal ➔ Modelization of the property
(e.g., Unlinkability)

Equivalence of two models

Automatic Verification

Secure
Big Picture

Protocol's specification → Protocol's model (e.g., ProVerif model)

Privacy goal (e.g., Unlinkability) → Modelization of the property

\( !\nu \text{ id} !\nu \text{ Sess. } P \approx !\nu \text{ id. } \nu \text{ Sess. } P \)

Theorem: two conditions \( \Rightarrow \) Unlinkability & Anonymity

Lucca Hirschi
A Method for Verifying Privacy-Type Properties: The Unbounded Case
I: Model & Problem
Any $\Sigma$-algebra + equational theory $E$ + reduction rules (à la Proverif)

**Example**

- $\Sigma_c = \{\text{dh}/2, \langle _, _ \rangle/2, \text{enc}/2, \text{ok}/0, \text{no}/0\}$
- $\Sigma_d = \{\pi_1/1, \pi_2/1, \text{dec}/2\}$
- $E = \{(\text{dh}(\text{dh}(x, y), z) = \text{dh}(\text{dh}(x, z), y))\}$
- $\text{def}_{\Sigma}(\text{dec}) = \{\text{dec}(\text{enc}(x, y), y) \rightarrow x\}$
- $\text{def}_{\Sigma}(\pi_i) = \{\pi_i(\langle x_1, x_2 \rangle) \rightarrow x_i\}$

**induce**

- a congruence $=_E$
  
  *e.g.*, $g^{xyz} =_E g^{zyx}$

- a “computation” relation $\Downarrow$
  
  *e.g.*, $\text{dec}(\text{enc}(n, g^{ab}), g^{ba}) \Downarrow n$
Applied-$\pi$ - Terms

Any $\Sigma$-algebra + equational theory $E$ + reduction rules (à la Proverif)

Example

$\Sigma_c = \{\text{dh}/2, \langle \_, \_ \rangle/2, \text{enc}/2, \text{ok}/0, \text{no}/0\}$

$\Sigma_d = \{\pi_1/1, \pi_2/1, \text{dec}/2\}$

$E = \{(\text{dh}(\text{dh}(x, y), z) = \text{dh}(\text{dh}(x, z), y))\}$

$\text{def}_\Sigma(\text{dec}) = \{\text{dec}(\text{enc}(x, y), y) \rightarrow x\}$

$\text{def}_\Sigma(\pi_i) = \{\pi_i(\langle x_1, x_2 \rangle) \rightarrow x_i\}$

induce

- a congruence $=_E$
  
  e.g., $g^{xyz} =_E g^{zyx}$

- a “computation” relation $\Downarrow$
  
  e.g., $\text{dec}(\text{enc}(n, g^{ab}), g^{ba}) \Downarrow n$

~~ We deal with arbitrary term algebra
Applied-\(\pi\) - Syntax

- Process: \(P, Q := \)
  - \(0\) null
  - \(\text{in}(c, x).P\) input
  - \(\text{out}(c, u).P\) output
  - if Test then \(P\) else \(Q\) conditional
  - \(P | Q\) parallel
Applied-$\pi$ - Syntax

- **Process:** $P, Q :=$
  - $0$ null
  - $\text{in}(c, x).P$ input
  - $\text{out}(c, u).P$ output
  - if Test then $P$ else $Q$ conditional
  - $P \mid Q$ parallel
  - $!P$ replication
  - $\nu n.P$ restriction

- Frame ($\phi$): the set of messages revealed to the attacker's knowledge
  - $\phi = \{w_1 \mapsto \text{enc}(m, k), w_2 \mapsto k\}$

- Configuration: $A = (P; \phi)$
Applied-$\pi$ - Syntax

- **Process:** $P, Q :=

  - $0$ null
  - $\text{in}(c, x).P$ input
  - $\text{out}(c, u).P$ output
  - if Test then $P$ else $Q$ conditional
  - $P \mid Q$ parallel
  - $!P$ replication
  - $\nu n. P$ restriction

- **Frame** ($\phi$): the set of messages revealed to the attacker.

  - intuition: attacker’s knowledge

  $$\phi = \{ w_1 \mapsto \text{enc}(m, k); w_2 \mapsto k \}$$

  - handle
  - out. message
Applied-$\pi$ - Syntax

- **Process:** $P, Q \ ::= \begin{align*}
0 & \quad \text{null} \\
\text{in}(c, x).P & \quad \text{input} \\
\text{out}(c, u).P & \quad \text{output} \\
\text{if Test then } P \ 	ext{else } Q & \quad \text{conditional} \\
P | Q & \quad \text{parallel} \\
!P & \quad \text{replication} \\
\nu n.P & \quad \text{restriction}
\end{align*}$

- **Frame** ($\phi$): the set of messages revealed to the attacker's knowledge

  Intuition: $\phi = \{ w_1 \mapsto \text{enc}(m, k); w_2 \mapsto k \}$

- **Configuration:** $A = (P; \phi)$
Applied-$\pi$ - Semantics

- **Recipes**: are terms built using handles
  
  \[ R = \text{dec}(w_1, w_2) \]
  \[ R\phi \Downarrow m \]

  for \( \phi = \{ w_1 \mapsto \text{enc}(m, k), w_2 \mapsto k \} \)

  “How \(\mathcal{R}\) builds messages from its knowledge”
Applied-$\pi$ - Semantics

- Recipes: are terms built using handles
  
  \[ R = \text{dec}(w_1, w_2) \]
  
  \[ R\phi \Downarrow m \]
  
  for \( \phi = \{w_1 \mapsto \text{enc}(m, k), w_2 \mapsto k\} \)

  “How 🤖 builds messages from its knowledge”

- Semantics of configurations:
  
  - Protocol’s output:
    
    \[ \text{Protocol’s output:} \]
    
    \[ (\{\text{out}(c, u).P\} \cup P; \phi) \xrightarrow{\text{out}(c, w)} (\{P\} \cup P; \phi \cup \{w \mapsto u\}) \]
    
    if \( w \) fresh

    🤖 learns outputted message
Applied-$\pi$ - Semantics

- Recipes: are terms built using handles

\[ R = \text{dec}(w_1, w_2) \]
\[ R\phi \Downarrow m \quad \text{for } \phi = \{ w_1 \mapsto \text{enc}(m, k), w_2 \mapsto k \} \]

"How \(\mathcal{M}\) builds messages from its knowledge"

- Semantics of configurations:
  - Protocol's output:
    \[
    (\{\text{out}(c, u).P\} \cup \mathcal{P}; \phi) \xrightarrow{\text{out}(c,w)} (\{P\} \cup \mathcal{P}; \phi \cup \{w \mapsto u\}) \quad \text{if } w \text{ fresh}
    \]
    \(\mathcal{M}\) learns outputted message
  - Protocol's input:
    \[
    (\{\text{in}(c, x).P\} \cup \mathcal{P}; \phi) \xrightarrow{\text{in}(c,R)} (\{P\{x \mapsto u\}\} \cup \mathcal{P}; \phi) \quad \text{if } R\phi \Downarrow u
    \]
    \(\mathcal{M}\) injects any message he can builds
Applied-$\pi$ - Semantics

> **Recipes**: are terms built using handles

\[ R = \text{dec}(w_1, w_2) \quad \text{for } \phi = \{ w_1 \mapsto \text{enc}(m, k), w_2 \mapsto k \} \]

“How \( \text{ alcançador} \) builds messages from its knowledge”

> **Semantics** of configurations:
- Protocol’s output:

\[
\begin{align*}
\{\text{out}(c, u).P\} \cup P; \phi & \xrightarrow{\text{out}(c,w)} \{P\} \cup P; \phi \cup \{w \mapsto u\} \\
\text{\( \text{ alcançador} \)} & \text{ learns outputted message}
\end{align*}
\]

- Protocol’s input:

\[
\begin{align*}
\{\text{in}(c, x).P\} \cup P; \phi & \xrightarrow{\text{in}(c,R)} \{P\{x \mapsto u\}\} \cup P; \phi \\
\text{\( \text{ alcançador} \)} & \text{ injects any message he can builds}
\end{align*}
\]

+ expected rules for conditional and other constructs

\( \mapsto \text{ alcançador} \) **controls all the network**
Applied-$\pi$ - Trace Equivalence

Unlinkability and Anonymity rely on trace equivalence
Applied-$\pi$ - Trace Equivalence

Unlinkability and Anonymity rely on trace equivalence

**Static Equivalence (intuitively)**

$\Phi \sim \Psi$ when
- $\text{dom}(\Phi) = \text{dom}(\Psi)$ and
- for all tests, it holds on $\phi \iff$ it holds on $\psi$
Applied-\( \pi \) - Trace Equivalence

Unlinkability and Anonymity rely on trace equivalence

**Static Equivalence (intuitively)**

\[ \Phi \sim \Psi \text{ when} \]

- \( \text{dom}(\Phi) = \text{dom}(\Psi) \) and
- for all tests, it holds on \( \phi \iff \) it holds on \( \psi \)

**Trace Equivalence**

\[ A \sqsubseteq B: \text{ for any } A \xrightarrow{\text{tr}} A' \text{ there exists } B \xrightarrow{\text{tr}} B' \text{ such that } \Phi(A') \sim \Phi(B'). \]

\[ A \approx B, \text{ when } A \sqsubseteq B \text{ and } B \sqsubseteq A. \]

- Intuition of \( A \sqsubseteq B \):
  - \( \forall \) and behaviour of \((A||)\) producing observable \( D \)
  - \( \Rightarrow \exists \) behaviour of \((B||)\) producing observable \( D' \sim D \)
I : Model & Problem
Unlinkability

Scenario 1 | Scenario 2

"Real" usage of the protocol | "Ideal" usage of the protocol

∀ ✋, ✋ cannot observe any difference
Unlinkability

Scenario 1  
"Real" usage of the protocol

≈

Scenario 2  
"Ideal" usage of the protocol

(≈: trace equivalence)
Unlinkability

Scenario 1

Session 1
Session 2
Session 3

Scenario 2

Session 1
Session 2
Session 3

→ Infinitely many users
→ Each playing infinitely many sessions
Unlinkability

Scenario 1

id_1
id_2
id_3

Session 1
Session 2
Session 3

Scenario 2

id_1
id_2
id_3

Session 1
Session 1
Session 1

∞ users
∞ sessions

!ν id !ν Sess. P
≈

∞ users

1 session

!ν id . ν Sess. P

(Strong unlinkability [Arapinis, Chothia, Ritter, Ryan CSF’10])
Goal

Automatic verification of

\[ \nu \text{ id.} (\nu \text{ Sess.} P) \approx \nu \text{ id.} (\nu \text{ Sess.} P) \]

for a large class of 2-party protocols (think of \( P = \text{Tag} \mid \text{Reader} \))
The Problem

Goal

Automatic verification of

\[
\nu \text{id.}(\nu \text{Sess.}P) \approx \nu \text{id.}(\nu \text{Sess.}P)
\]

for a large class of 2-party protocols (think of \( P = \text{Tag} | \text{Reader} \))

Our class of protocols

- Intuitively, a party \( P \) is a process of the form:

\[
\begin{align*}
P &::= 0 \mid \text{in}(c, y). \text{if Test then out}(c, u).P \text{ else } P_{\text{else}} \\
P_{\text{else}} &::= 0 \mid \text{out}(c', u')
\end{align*}
\]
The Problem

Goal

Automatic verification of

\[
\nu \ id. (\nu \ Sess. P) \approx \nu \ id. (\nu \ Sess. P)
\]

\(\mathcal{M}\) \(\mathcal{S}\)

for a large class of 2-party protocols (think of \(P = \text{Tag} \mid \text{Reader}\))

Our class of protocols

- Intuitively, a party \(P\) is a process of the form:

\[
P ::= 0 \mid \text{in}(c, y). \text{if} \ Test \text{then} \text{out}(c, u). P \text{ else } P_{\text{else}}
\]

\[
P_{\text{else}} ::= 0 \mid \text{out}(c', u')
\]

- A protocol \(\Pi\) is a tuple \((\vec{k}, \vec{n}_T, \vec{n}_R, T, R)\) where:
  - \(T\) and \(R\) are parties
  - \(\vec{k}\): identity names and \(\vec{n}_T / \vec{n}_R\): session names
  - \(\text{fn}(T) \subseteq \vec{k} \sqcup \vec{n}_T\) (resp. for \(R\))
Existing Approaches

Goal

Automatic verification of

\[
\nu id. (\nu Sess.P) \approx \nu id. (\nu Sess.P)
\]

for a large class of 2-party protocols (think of \( P = \text{Tag} | \text{Reader} \))

Existing approaches:

- **manual**: long, difficult, and highly error prone
- **automatic** (only ProVerif/Maude-NPA/Tamarin):
  - rely on too imprecise approximation of \( \approx \): diff-equivalence
  - \( \leftrightarrow \) always fail to prove unlinkability
Contributions

Theory:
- 2 reasonable conditions implying unlinkability (& anonymity)
- for a large class of 2-party protocols
Contributions

Theory:
- 2 reasonable conditions implyingunlinkability (& anonymity)
- for a large class of 2-party protocols

Practice:
- our conditions can be checked automatically using existing tools
- we provide tool support for that (UKano)
Contributions

Theory:
▶ 2 reasonable conditions implying unlinkability (& anonymity)
▶ for a large class of 2-party protocols

Practice:
▶ our conditions can be checked automatically using existing tools
▶ we provide tool support for that (UKano)

Applications:
▶ new proofs & attacks on RFID protocols
Outline

I Model & Problem

II Sufficient Conditions

III Mechanization & Applications

IV Conclusion
II: Two Generic Classes of Attacks
Two Conditions to Avoid them
1st Class: Leaks through Relations over Messages

![Diagram of Tag Reader relationship]

Tag: \( k, id \)
Reader: \( k \)

\( \{id\}_k \)
1st Class:Leaks through Relations over Messages

\[ \text{Tag}_1 \]
\[ k_1, id_1 \]
\[ \{id_1\}_{k_1} \]

\[ \text{Tag}_2 \]
\[ k_2, id_2 \]
\[ \{id_2\}_{k_2} \]

\[ \Rightarrow \]

?
1st Class:Leaks through Relations over Messages

\[ (k_1, id_1) = (k_2, id_2) \]
1\textsuperscript{st} Class:Leaks through Relations over Messages

\begin{align*}
\text{Tag}_1 & \quad k_1, \text{id}_1 \\
\{\text{id}_1\}_{k_1} & \\
\text{Tag}_2 & \quad k_2, \text{id}_2 \\
\{\text{id}_2\}_{k_2} & \\
(k_1, \text{id}_1) & \neq (k_2, \text{id}_2)
\end{align*}
1st Class: Leaks through Relations over Messages
1\textsuperscript{st} Class: Leaks through Relations over Messages

\textbf{Tag}_1

\begin{align*}
  k_1, \text{id}_1 & \quad \rightarrow & 0 \\
  \{0, \text{id}_1\}_{k_1} & \quad \rightarrow & \text{Tag}_2
\end{align*}

\textbf{Tag}_2

\begin{align*}
  k_2, \text{id}_2 & \quad \rightarrow & 0 \\
  \{0, \text{id}_2\}_{k_2} & \quad \rightarrow & \quad \Rightarrow \quad ?
\end{align*}
1st Class: Leaks through Relations over Messages

Problem

For some 🐻’s behaviors, relations over messages leak info about involved agents.
1st Class: Leaks through Relations over Messages

Problem

For some 🎶’s behaviors, relations over messages leak info about involved agents.

Main idea to avoid that:

▶ no relation at all
▶ (roughly) ⇝ outputs are indistinguishable from fresh nonces
Problem

For some $\mathcal{B}$'s behaviors, relations over messages leak info about involved agents.

Main idea to avoid that:

- no relation at all
- (roughly) $\sim$ outputs are indistinguishable from fresh nonces

$1^{st}$ Condition: Frame Opacity (informal)

For all $\mathcal{M} \xrightarrow{t} (\mathcal{P}; \Phi)$, we have that $\Phi \sim [\Phi]^\text{nonce}$. 

"real" frame  "ideal" frame

[Φ]nonce
Problem

For some agent's behaviors, relations over messages leak info about involved agents.

Main idea to avoid that:
- no relation at all
- (roughly) $\sim$ outputs are indistinguishable from fresh nonces

1st Condition: Frame Opacity (informal)

For all $M \xrightarrow{t} (P; \Phi)$, we have that $\Phi \sim [\Phi]^{nonce}$.

- $\Phi = \{ w \mapsto \langle \text{enc}(n_1, k), \text{enc}(n_2, k) \rangle \}$
- $[\Phi]^{nonce} = \{ w \mapsto \langle n, n' \rangle \}$ and $\Phi \sim [\Phi]^{nonce}$

$[\Phi]^{nonce}$ based on a notion of transparent function symbols
2nd Class: Leaks through Conditionals’ Outcomes

Lucca Hirschi  A Method for Verifying Privacy-Type Properties: The Unbounded Case 20 / 29
2nd Class: Leaks through Conditionals’ Outcomes

Lucca Hirschi
A Method for Verifying Privacy-Type Properties: The Unbounded Case
2nd Class: Leaks through Conditionals’ Outcomes

\[\text{Tag}_1 \xrightarrow{k_1} \{n_1\} \xleftarrow{k_1} \text{Reader}_1\]

\[\text{Tag}_2 \xrightarrow{k_2} \{n_2\} \xleftarrow{k_2} \text{Reader}_2\]
2\textsuperscript{nd} Class: Leaks through Conditionals’ Outcomes

\begin{itemize}
\item Tag\textsubscript{1} $k_1$
\item Reader\textsubscript{1} $k_1$
\item Tag\textsubscript{2} $k_2$
\item Reader\textsubscript{2} $k_2$
\end{itemize}

\begin{align*}
\text{dec}(X, k_2) &\neq \perp \\
\text{if } \text{dec}(X, k_2) &\neq \perp
\end{align*}
2nd Class: Leaks through Conditionals’ Outcomes

\[ k_1 = k_2 \]

\[ \text{if } \text{dec}(X, k_2) \neq \bot \]

\[ \{n_1\}_{k_1} \]

\[ \{n_2\}_{k_2} \]

\[ \{n\}'_{k_2} \]
2\textsuperscript{nd} Class: Leaks through Conditionals’ Outcomes

\[ \text{Tag}_1 \quad \text{Reader}_1 \]

\[ \begin{array}{c}
\{ n_1 \}_{k_1} \\
\text{..} \\
\end{array} \quad \text{..} \quad \begin{array}{c}
\text{..} \\
\{ n_1 \}_{k_1} \\
\end{array} \]

\[ \text{Tag}_2 \quad \text{Reader}_2 \]

\[ \begin{array}{c}
\{ n_2 \}_{k_2} \\
\text{..} \\
\end{array} \quad \text{..} \quad \begin{array}{c}
\{ n_1 \}_{k_1} \\
\text{..} \\
\end{array} \]

\[ \text{if dec}(X, k_2) \neq \perp \]

\[ \Rightarrow k_1 \neq k_2 \]
Problem

For some 🤔’s behaviors, conditionals’ outcomes leak info about involved agents.

Main idea to avoid that:

- when 🤔 plays such an attack ⇒ conditional evaluates negatively
- ⇔ conditional evaluates positively ⇐⇒ 🤔 did not interfer
2nd Class: Leaks through Conditionals’ Outcomes

Problem

For some 🙁’s behaviors, conditionals’ outcomes leak info about involved agents.

Main idea to avoid that:

- when 🙁 plays such an attack ⇒ conditional evaluates negatively
- ⇔ conditional evaluates positively ⇔ 🙁 did not interfere

2nd Condition: Well-Authentication (informal)

\[ \forall M \quad t.test-\text{ok}[T(id, sess)] \rightarrow (P; \Phi) \]

there must be a \( R(id, sess') \) such that \( T(id, sess) \) and \( R(id, sess') \) were having an honest interaction.
Main Result

Theorem

For any protocol in our class:

\[
\begin{align*}
\text{frame opacity} \\
& \& \\
\text{well-authentication}
\end{align*}
\] \Rightarrow

\[
\begin{align*}
\text{Unlinkability} \\
& \& \\
\text{Anonymity}
\end{align*}
\]
III : Mechanization & Applications
Mechanization

Both conditions can be automatically verified using ProVerif:

- **Well Authentication:** $\equiv$ just reachability properties
  - no longer equivalence property

Tool: UKano

Built on top of ProVerif that automatically checks our conditions.
Sources of UKano at http://projects.lsv.ens-cachan.fr/ukano/
Mechanization

Both conditions can be automatically verified using ProVerif:

- **Well Authentication:** ~⇒ just reachability properties
  - no longer equivalence property

- **Frame Opacity:** ~⇒ equivalence between messages
  - checkable with good precision via diff-equivalence and encodings
Mechanization

Both conditions can be automatically verified using ProVerif:

- **Well Authentication:** $\rightsquigarrow$ just reachability properties
  - no longer equivalence property

- **Frame Opacity:** $\rightsquigarrow$ equivalence between messages
  - checkable with good precision via diff-equivalence and encodings

**Tool: UKano**

Built on top of ProVerif that **automatically checks** our conditions.

Sources of UKano at

http://projects.lsv.ens-cachan.fr/ukano/
Modelization of the property $\nu_{id} \nu_{Sess.} P \approx \nu_{id}.\nu_{Sess.} P$

Equivalence of two models

ProVerif

Protocol's specification

Protocol's model (ProVerif model)

Secure

No

Unlinkability or/and Anonymity

Modelization of the property

Well-Authentication

Frame Opacity

Attack
## Case Studies

<table>
<thead>
<tr>
<th>RFID auth. protocol</th>
<th>Frame opacity</th>
<th>Well-auth.</th>
<th>Unlinkability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feldhofer</td>
<td>✓</td>
<td>✓</td>
<td>safe</td>
</tr>
<tr>
<td>Hash-Lock</td>
<td>✓</td>
<td>✓</td>
<td>safe</td>
</tr>
<tr>
<td>LAK (stateless)</td>
<td>–</td>
<td>✗</td>
<td>🎱</td>
</tr>
<tr>
<td>Fixed LAK</td>
<td>✓</td>
<td>✓</td>
<td>safe</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ePassport protocol</th>
<th>Frame opacity</th>
<th>Well-auth.</th>
<th>Unlinkability</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAC</td>
<td>✓</td>
<td>✓</td>
<td>safe</td>
</tr>
<tr>
<td>BAC/PA/AA</td>
<td>✓</td>
<td>✓</td>
<td>safe</td>
</tr>
<tr>
<td>PACE (faillible dec)</td>
<td>–</td>
<td>✗</td>
<td>🎱</td>
</tr>
<tr>
<td>PACE (missing test)</td>
<td>–</td>
<td>✗</td>
<td>🎱</td>
</tr>
<tr>
<td>PACE</td>
<td>–</td>
<td>✗</td>
<td>🎱</td>
</tr>
<tr>
<td>PACE with tags</td>
<td>✓</td>
<td>✓</td>
<td>safe</td>
</tr>
</tbody>
</table>

- Found **automatically new proofs and new attacks** using UKano
IV : Conclusion
Conclusion

- **Theory**: 2 conditions $\Rightarrow$ unlinkability & anonymity
- **Practice**: UKano automatically verifies them
- **Applications**: new proofs & attacks on RFID protocols
Conclusion

- **Theory**: 2 conditions ⇒ unlinkability & anonymity
- **Practice**: UKano automatically verifies them
- **Applications**: new proofs & attacks on RFID protocols

### Future Current Work:

- more precise notion of **frame opacity** (for e.g., signature, ZK)
- **extend** the **class** of protocols + unlinkability **scenarios**
  - DAA, ABCDH: new case studies
- for standard crypto, conditions checkable **without equivalence**
Conclusion

Future Work

Improve the method:
- tackle memory (often used in RFID)
- move to other tools as backends (Tamarin, Maude-NPA)
- allow more flexibility for honest interactions

Reusing core ideas:
- reuse methodology for other contexts/privacy properties
e-voting: done for ballot secrecy with Cas Cremers this summer
- extract guidelines for privacy from our conditions
Conclusion

Future Work

Improve the method:
- tackle memory (often used in RFID)
- move to other tools as backends (Tamarin, Maude-NPA)
- allow more flexibility for honest interactions

Reusing core ideas:
- reuse methodology for other contexts/privacy properties
  - e-voting: done for ballot secrecy with Cas Cremers this summer
  - (?) attribute-based credentials, TPM, blockchain technologies, transparent certificate authorities, …
- extract guidelines for privacy from our conditions
Paper, sources of UKano, ProVerif files at
http://projects.lsv.ens-cachan.fr/ukano/

Thank you!
More: Extensions
Extensions

A more precise notion of **frame opacity** (for e.g., signature, ZK)
B **extend** the **class** of protocols + unlinkability **scenarios**
   - DAA, ABCDH: new case studies
C for standard crypto, conditions checkable **without equivalence**
Example DAA Sign:

\[ \text{TPM(cred)} \rightarrow \text{Verifier} \]

\[ n \]

\[ \text{ZK}(n, \text{cred}) \]
Example DAA Sign:

\[ \text{Verifier} \rightarrow n \rightarrow \text{TPM(cred)} \rightarrow \text{Verifier} \]

\[ \text{ZK}(n, \text{cred}) \]

\[ \text{fresh nonce} \]

Before

\[ \text{Relations must only depend on what is already observable} \]

For any execution

\[ M_{t} \rightarrow (P; \Phi) \]

we have that

\[ \Phi \sim \Phi(B) \text{ nonce} \]

Now

\[ \text{Relations must only depend on what is already observable} \]

For any execution

\[ M_{t} \rightarrow (P; \Phi) \]

we have that

\[ \Phi \sim \text{ideal}(t) \]
A: Improving Preciseness of Frame Opacity

Before

- No relation at all
- $\sim$ Outputs are indistinguishable from fresh nonces

For any execution $\mathcal{M} \xrightarrow{t} (\mathcal{P}, \Phi)$, we have that $\Phi \sim [\Phi(B)]_{\text{nonce}}$. 
A: Improving Preciseness of Frame Opacity

Before

- No relation at all
- \(\sim\) Outputs are indistinguishable from fresh nonces

For any execution \(\mathcal{M} \xrightarrow{t} (\mathcal{P}, \Phi)\), we have that \(\Phi \sim [\Phi(B)]^{\text{nonce}}\).

Now

- Relations must only depend on what is already observable

For any execution \(\mathcal{M} \xrightarrow{t} (\mathcal{P}; \Phi)\), we have that \(\Phi \sim \text{ideal}(t)\).
B: Variations of Scenarios

Unlinkability

\[ \begin{align*} &! \nu \text{id.} \! \nu \text{Sess.}(T | R) \approx ! \nu \text{id.} (\nu \text{Sess.}(T | R)) \end{align*} \]

Well-Authentication

\[ \forall (M; \emptyset) \xrightarrow{t.\text{then}[T(\text{id},\text{sess})]} (P; \Phi) \]

there must be a \( R(\text{id}, \text{sess}') \) such that \( T(\text{id}, \text{sess}) \) and \( R(\text{id}, \text{sess}') \) were having an honest interaction.
B: Variations of Scenarios

What if $R$ has no proper identity?

**Unlinkability**

$$(! \nu \text{id}_T. ! \nu \text{Sess}_T.T) | !\nu \text{Sess}_R.R \approx (! \nu \text{id}_T. \nu \text{Sess}_T.T) | !\nu \text{Sess}_R.R$$

**Well-Authentication**

$$\forall (M; \emptyset) \xrightarrow{t.\text{then}[T(\text{id}, \text{sess})]} (P; \Phi)$$

there must be a $R(\text{sess}')$ such that $T(\text{id}, \text{sess})$ and $R(\text{sess}')$ were having an honest interaction.
B: Variations of Scenarios

What if $T$ and $R$ never share identity?

### Unlinkability

$$(! \nu \text{id}_T. ! \nu \text{Sess}_T.T) \mid (! \nu \text{id}_R. ! \nu \text{Sess}_R.R)$$

$$\approx$$

$$(! \nu \text{id}_T. \nu \text{Sess}_T.T) \mid (! \nu \text{id}_R. \nu \text{Sess}_R.R)$$

### Well-Authentication

$$\forall (\mathcal{M}; \emptyset) \xrightarrow{\text{t.then}[T(\text{id, sess})]} (\mathcal{P}; \Phi)$$

there must be a $R(\text{id}', \text{sess}')$ such that $T(\text{id, sess})$ and $R(\text{id}', \text{sess}')$ were having an honest interaction.
What if sessions of $T$ (or/and $R$) cannot be executed concurrently?

**Unlinkability**

$$!\nu\text{id. } i\nu\text{ Sess.}(T | R) \approx !\nu\text{id. } (\nu\text{ Sess.}(T | R))$$

$$!P \sim P | P | P | \ldots$$

$$iP \sim P; P; P; \ldots$$
B: Variations of Scenarios

1. What if $R$ has no proper identity?
2. What if $T$ and $R$ never share identity?
3. What if sessions of $T$ (or $R$) cannot be executed concurrently?

- Now, we can deal with all combinations of those variations
- They all are over-approximations of strong unlinkability
- new case studies: DAA Join & Sign, attribute-based authentication (abcdh used in IRMA)

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Frame opacity</th>
<th>Well-auth.</th>
<th>Unlinkability</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAA sign</td>
<td>✓</td>
<td>✓</td>
<td>safe</td>
</tr>
<tr>
<td>DAA join</td>
<td>✓</td>
<td>✓</td>
<td>safe</td>
</tr>
<tr>
<td>abcdh (irma)</td>
<td>✓</td>
<td>✓</td>
<td>safe</td>
</tr>
</tbody>
</table>
For standard crypto (enc, hash, mac, sign, data structures):

- syntactical checks + secrecy checks $\Rightarrow$ Frame Opacity
- $\leadsto$ bunch of reachability checks $\Rightarrow$ UK & ANO
C: UK & Ano $\rightsquigarrow$ Reachability Problem

(More prospective: defs OK but proofs not finished yet.)

For standard crypto (enc, hash, mac, sign, data structures):

- syntactical checks + secrecy checks $\Rightarrow$ Frame Opacity
- $\rightsquigarrow$ bunch of reachability checks $\Rightarrow$ UK & ANO

Sufficient heuristic: All top-most “crypto” messages:

- (a) cannot be forged by 😈. Checked via secrecy of keys/sub-messages.
- (b) are pairwise distinct. Checked via freshness syntactical conditions.
C: UK & Ano \(\rightsquigarrow\) Reachability Problem

(More propsective: defs OK but proofs not finished yet.)

For standard crypto (enc, hash, mac, sign, data structures):
  - syntactical checks + secrecy checks \(\Rightarrow\) Frame Opacity
  - \(\rightsquigarrow\) bunch of reachability checks \(\Rightarrow\) UK & ANO

Sufficient heuristic: All top-most “crypto” messages:
  - (a) cannot be forged by \(\ominus\). Checked via secrecy of keys/sub-messages.
  - (b) are pairwise distinct. Checked via freshness syntactical conditions.

Those messages are “black-boxed”: without any relation
\(\Rightarrow\) Frame Opacity