Can Charlie distinguish Alice and Bob?

Automated verification of equivalence properties

Steve Kremer

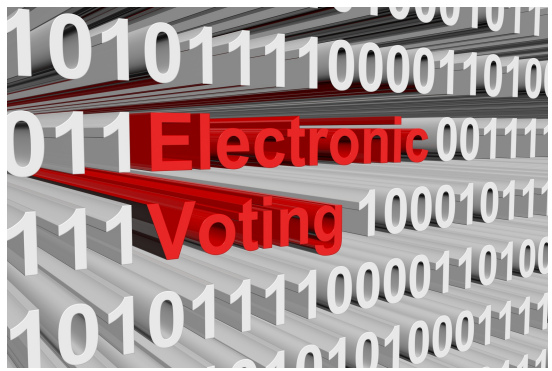
joint work with:

Myrto Arapinis, David Baelde, Rohit Chadha, Vincent Cheval, Ştefan Ciobâcâ, Véronique Cortier, Stéphanie Delaune, Ivan Gazeau, Itsaka Rakotonirina, Mark Ryan

29th IEEE Computer Security Foundations Symposium

Cryptographic protocols everywhere!

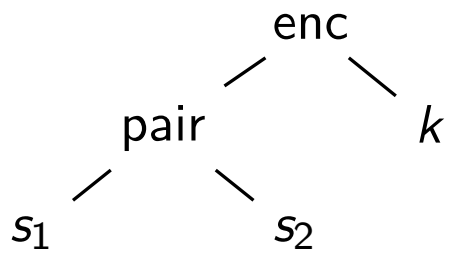
- Distributed programs that
- use crypto primitives (encryption, digital signature ,. . .)
- to ensure security properties (confidentiality, authentication, anonymity,. . .)



Symbolic models for protocol verification

Main ingredient of symbolic models

- messages = **terms**



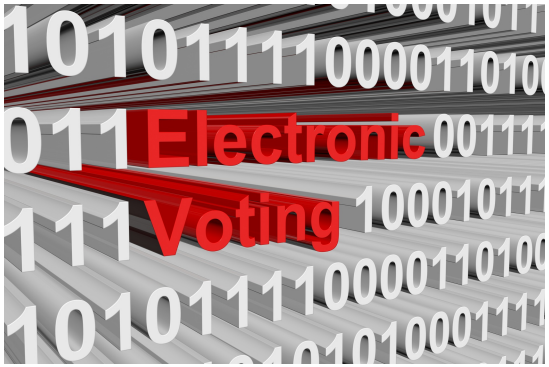
- **perfect** cryptography (equational theories)

$$\mathsf{dec}(\mathsf{enc}(x,y),y) = x \quad \mathsf{fst}(\mathsf{pair}(x,y)) = x \quad \mathsf{snd}(\mathsf{pair}(x,y)) = y$$

- the network **is** the attacker
  - messages can be eavesdropped
  - messages can be intercepted
  - messages can be injected

Dolev, Yao: On the Security of Public Key Protocols. FOCS'81

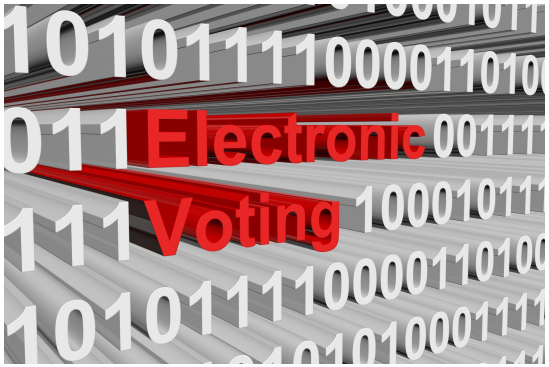
Cryptographic protocols are tricky!



Cryptographic protocols are tricky!

Bhargavan et al.:FREAK, Logjam, SLOTH, ...

Cremers et al., S&P'16

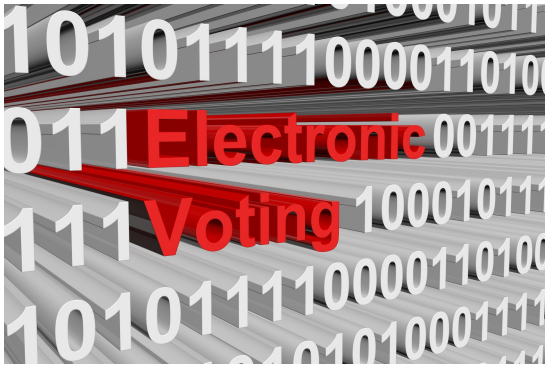


Cryptographic protocols are tricky!

Bhargavan et al.:FREAK, Logjam, SLOTH, ...

Cremers et al., S&P'16

Arapinis et al., CCS'12

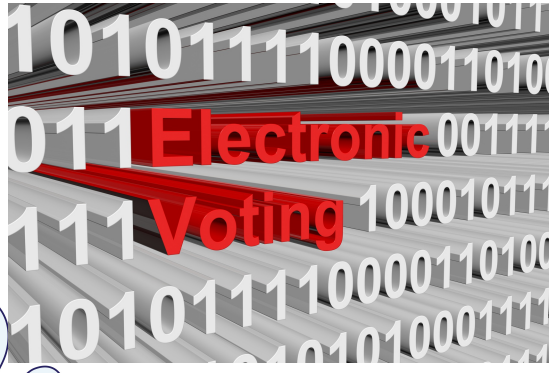


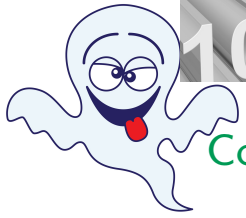
Cryptographic protocols are tricky!

Bhargavan et al.:FREAK, Logjam, SLOTH, ...

Cremers et al., S&P'16

Arapinis et al., CCS'12





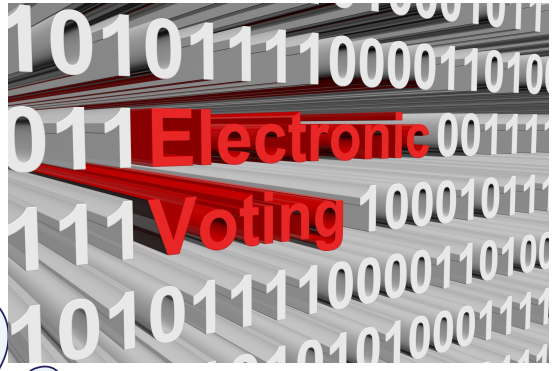
Cortier & Smyth, CSF'11

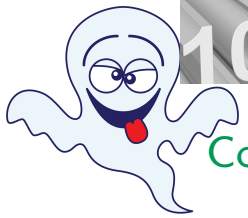
Cryptographic protocols are tricky!

Bhargavan et al.:FREAK, Logjam, SLOTH, . . .

Cremers et al., S&P'16

Arapinis et al., CCS'12





Cortier & Smyth, CSF'11

Steel et al., CSF'08, CCS'10

Modelling the protocol

Protocols modelled in a process calculus, e.g. the applied pi calculus

$$
\begin{array}{lll}
P ::= & 0 & \\
\mid & \mathsf{in}(c, x).P & \text{input} \\
\mid & \mathsf{out}(c, t).P & \text{output} \\
\mid & \text{if } t_1 = t_2 \text{ then } P \text{ else } Q & \text{conditional} \\
\mid & P \parallel Q & \text{parallel} \\
\mid & !P & \text{replication} \\
\mid & \text{new } n.P & \text{restriction}
\end{array}
$$

Modelling the protocol

Protocols modelled in a process calculus, e.g. the applied pi calculus

$$
\begin{array}{lll}
P ::= & 0 & \\
\mid & \mathsf{in}(c, x).P & \text{input} \\
\mid & \mathsf{out}(c, t).P & \text{output} \\
\mid & \text{if } t_1 = t_2 \text{ then } P \text{ else } Q & \text{conditional} \\
\mid & P \parallel Q & \text{parallel} \\
\mid & !P & \text{replication} \\
\mid & \text{new } n.P & \text{restriction}
\end{array}
$$

Specificities:

- messages are **terms** (not just names as in the pi calculus)
- equality in conditionals interpreted modulo an **equational theory**

Reasoning about attacker knowledge

Terms output by a process are organised in a **frame**:

$$\phi = \text{new } \bar{n}.\ \{^{t_1}/_{x_1}, \ldots, ^{t_n}/_{x_n}\}$$

Reasoning about attacker knowledge

Terms output by a process are organised in a **frame**:

$$\phi = \text{new } \bar{n}.\ \{^{t_1}/_{x_1}, \ldots, ^{t_n}/_{x_n}\}$$

Deducibility:

$\phi \vdash^R t$ if R is a public term and $R\phi =_E t$

Example

$$\varphi = \text{new } n_1, n_2, k_1, k_2.\ \{^{\text{enc}(n_1,k_1)}/_{x_1}, ^{\text{enc}(n_2,k_2)}/_{x_2}, ^{k_1}/_{x_3}\}$$

$$\varphi \vdash^{\text{dec}(x_1,x_3)} n_1 \qquad \varphi \nvdash n_2 \qquad \varphi \vdash^{\mathbf{1}} \mathbf{1}$$

Reasoning about attacker knowledge

Terms output by a process are organised in a **frame**:

$$\phi = \text{new } \bar{n}.\ \{^{t_1}/_{x_1}, \ldots, ^{t_n}/_{x_n}\}$$

Static equivalence:

$\phi_1 \sim_s \phi_2$ if $\forall$ public terms R, R'.

$$R\phi_1 = R'\phi_1 \Leftrightarrow R\phi_2 = R'\phi_2$$

Examples

$$\text{new } k.\ \{^{\text{enc}(\mathbf{0},k)}/_{x_1}\} \sim_s \text{new } k.\ \{^{\text{enc}(\mathbf{1},k)}/_{x_1}\}$$

Reasoning about attacker knowledge

Terms output by a process are organised in a **frame**:

$$\phi = \text{new } \bar{n}.\ \{^{t_1}/_{x_1}, \ldots, ^{t_n}/_{x_n}\}$$

Static equivalence:

$\phi_1 \sim_s \phi_2$ if $\forall$ public terms R, R'.

$$R\phi_1 = R'\phi_1 \Leftrightarrow R\phi_2 = R'\phi_2$$

Examples

$$\text{new } n_1, n_2.\ \{^{n_1}/_{x_1}, ^{n_2}/_{x_2}\} \not\sim_s \text{new } n_1, n_2.\ \{^{n_1}/_{x_1}, ^{n_1}/_{x_2}\}$$

Check $(x_1 \stackrel{?}{=} x_2)$

Reasoning about attacker knowledge

Terms output by a process are organised in a **frame**:

$$\phi = \text{new } \bar{n}.\ \{^{t_1}/_{x_1}, \ldots, ^{t_n}/_{x_n}\}$$

Static equivalence:

$\phi_1 \sim_s \phi_2$ if $\forall$ public terms R, R'.

$$R\phi_1 = R'\phi_1 \Leftrightarrow R\phi_2 = R'\phi_2$$

Examples

$$\{^{\text{enc}(n,k)}/_{x_1}, ^{k}/_{x_2}\} \not\sim_s \{^{\text{enc}(\mathbf{0},k)}/_{x_1}, ^{k}/_{x_2}\}$$

Check $(dec(x_1, x_2) \stackrel{?}{=} \mathbf{0})$

From authentication to privacy

Many good tools:

AVISPA, Casper, Maude-NPA, ProVerif, Scyther, Tamarin, ...

Good at verifying **trace properties** (predicates on system behavior), e.g.,

- (weak) secrecy of a key
- authentication (correspondence properties)

  If B ended a session with A (and parameters p) then A must have started a session with B (and parameters p').

From authentication to privacy

Many good tools:

AVISPA, Casper, Maude-NPA, ProVerif, Scyther, Tamarin, ...

Good at verifying **trace properties** (predicates on system behavior), e.g.,

- (weak) secrecy of a key
- authentication (correspondence properties)

  If B ended a session with A (and parameters p) then A must have started a session with B (and parameters p').

Not all properties can be expressed on a trace.

$\rightsquigarrow$ recent interest in **indistinguishability properties**.

Indistinguishability as a process equivalence

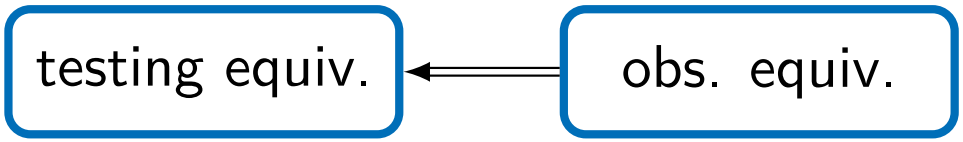
Naturally modelled using **equivalences** from process calculi

Testing equivalence ($P \approx Q$)
for all processes A, we have that:

$$A \mid P \Downarrow c \text{ if, and only if, } A \mid Q \Downarrow c$$

$\longrightarrow$ $P \Downarrow c$ when P can send a message on the channel c.

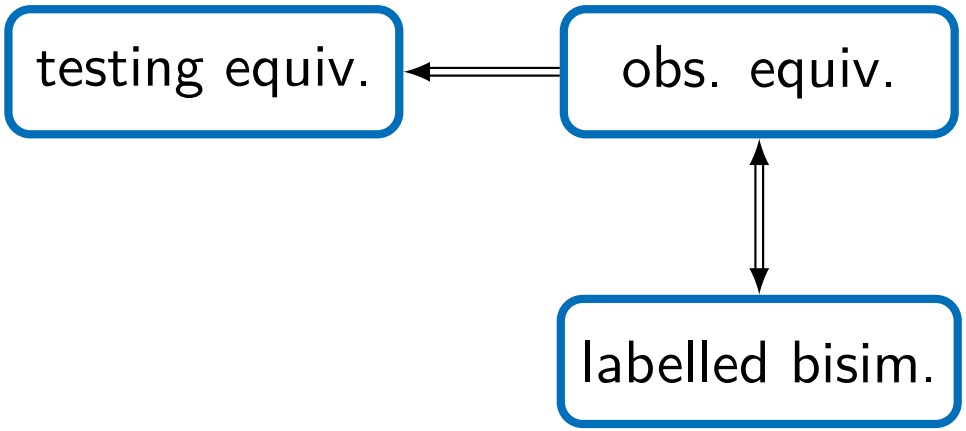
A tour to the (equivalence) zoo



Abadi, Gordon. A Calculus for Cryptographic Protocols: The Spi Calculus. CCS'97, Inf.& Comp.'99

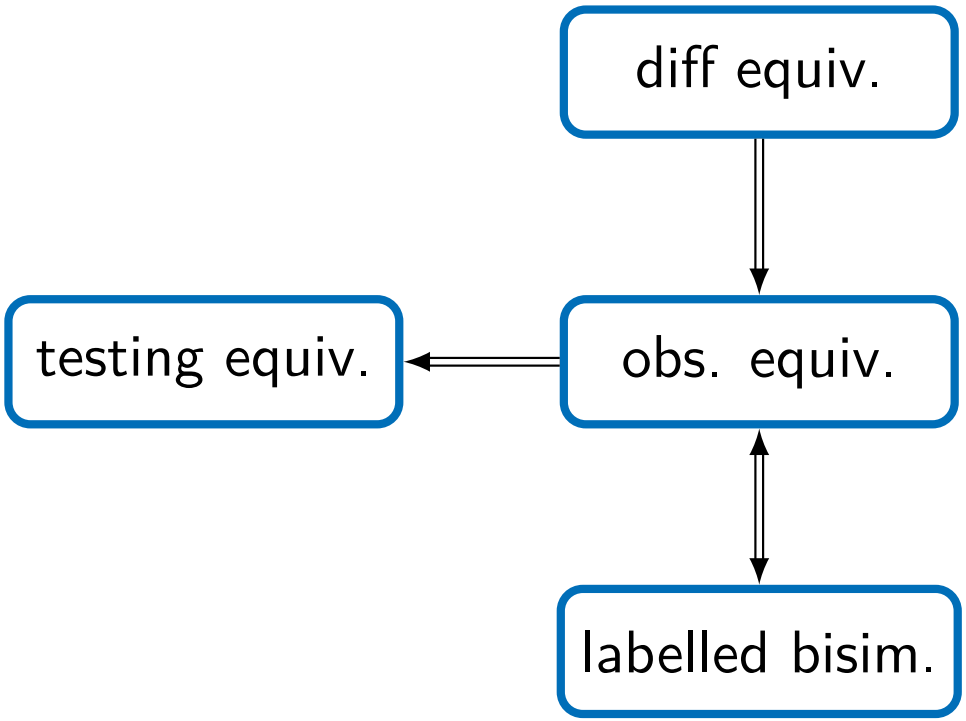
Abadi, Fournet. Mobile values, new names, and secure communication. POPL'01

A tour to the (equivalence) zoo



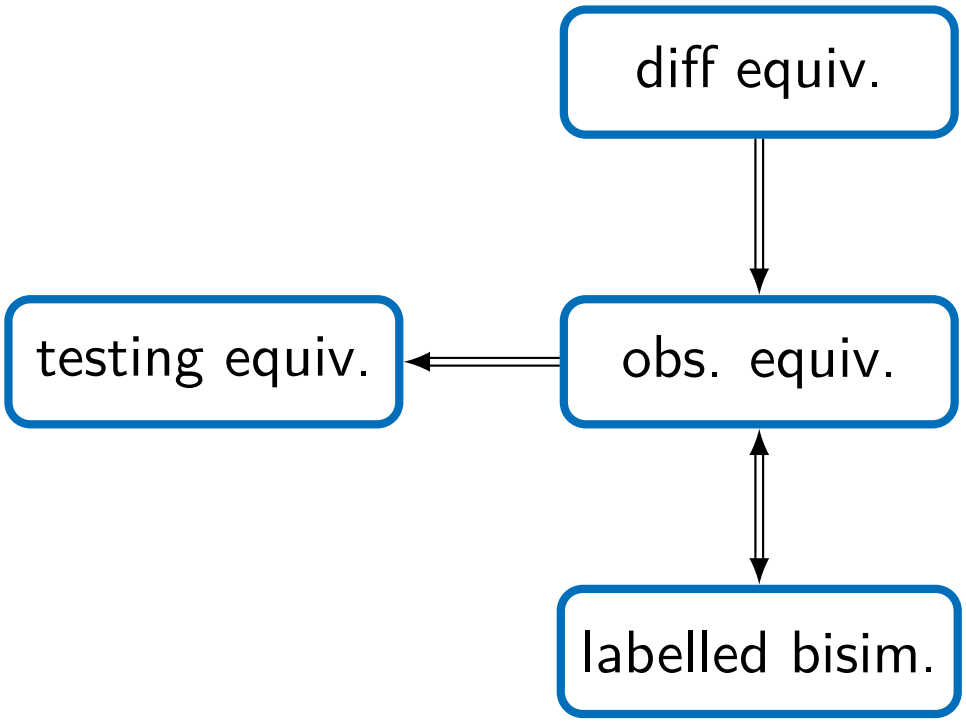
Abadi, Fournet. Mobile values, new names, and secure communication. POPL'01

A tour to the (equivalence) zoo



Blanchet et al.: Automated Verification of Selected Equivalences for Security Protocols. LICS'05

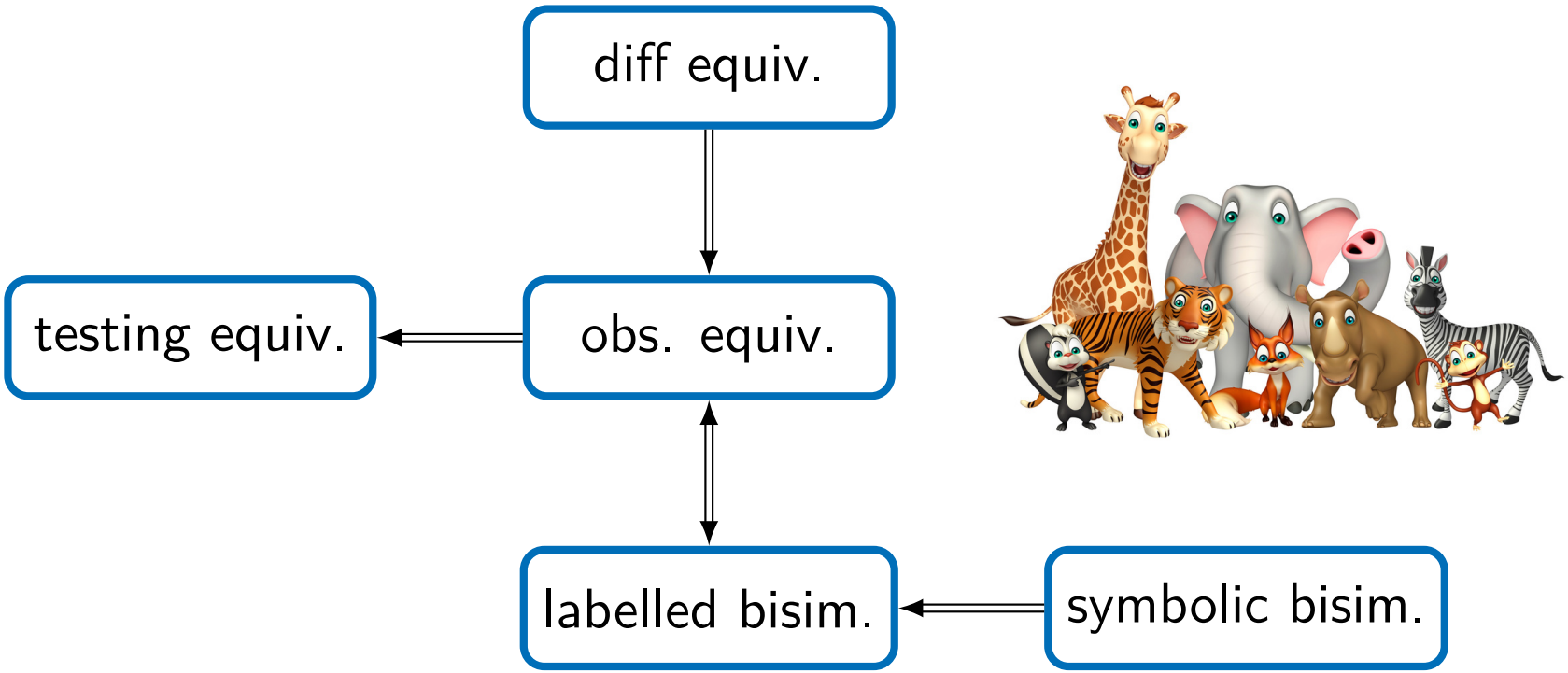
A tour to the (equivalence) zoo



Diff equivalence **too fine grained** for several properties.

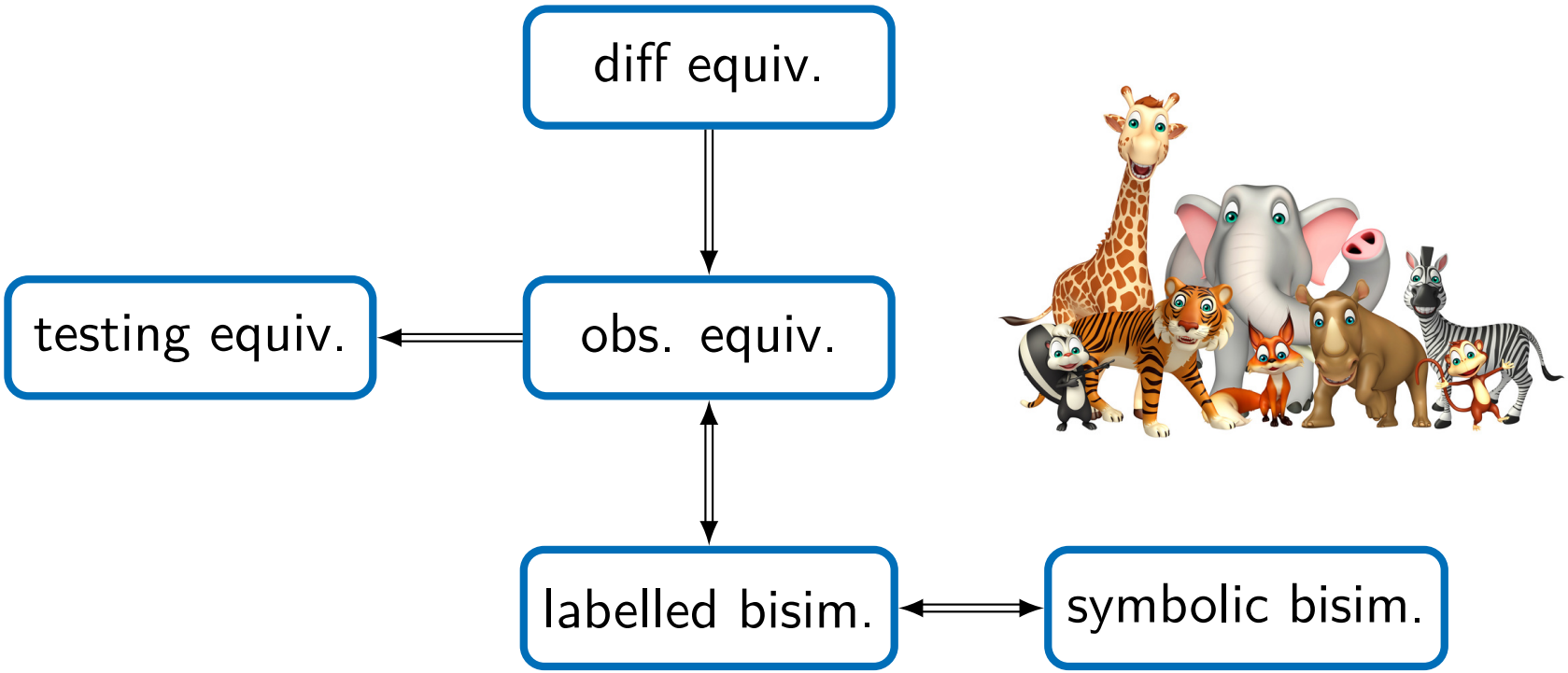
Blanchet et al.: Automated Verification of Selected Equivalences for Security Protocols. LICS'05

A tour to the (equivalence) zoo



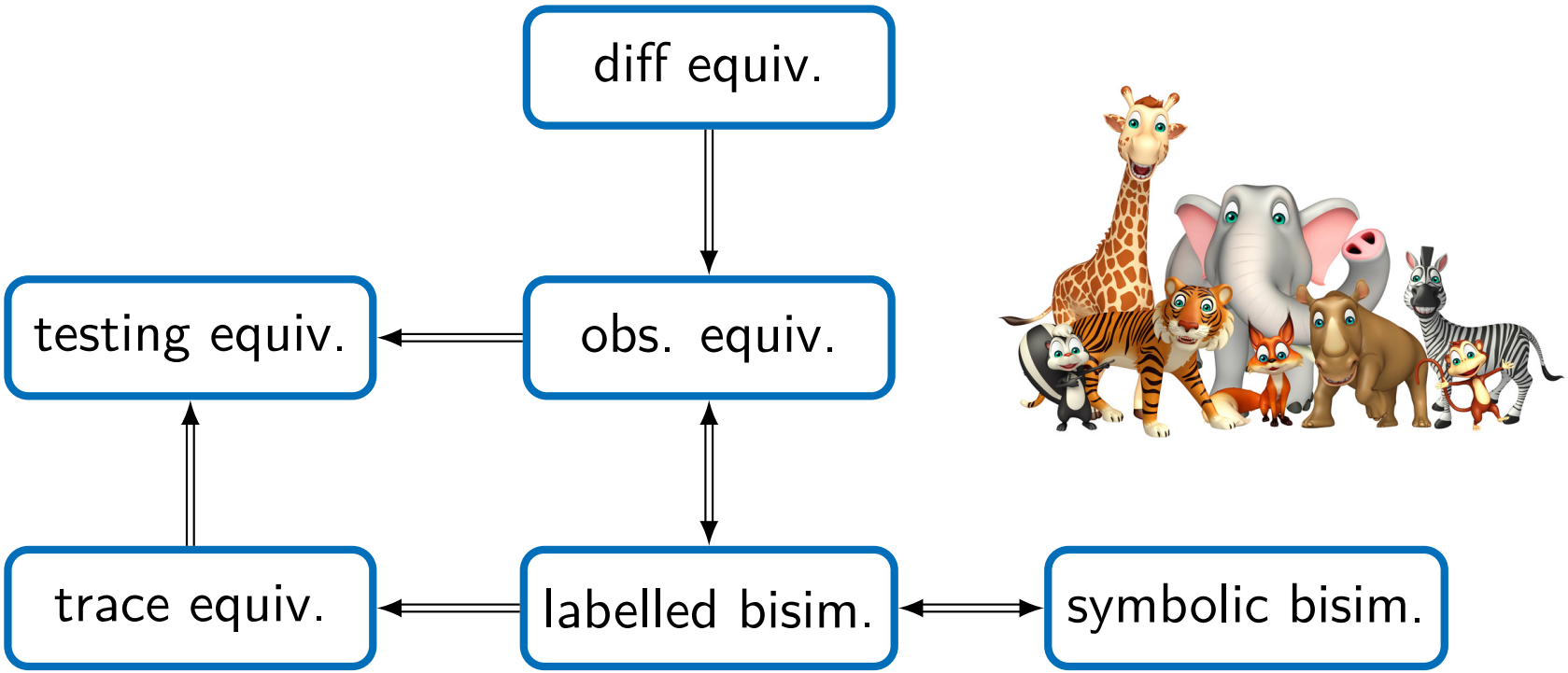
Delaune et al. Symbolic bisimulation for the applied pi calculus. JCS'10

A tour to the (equivalence) zoo



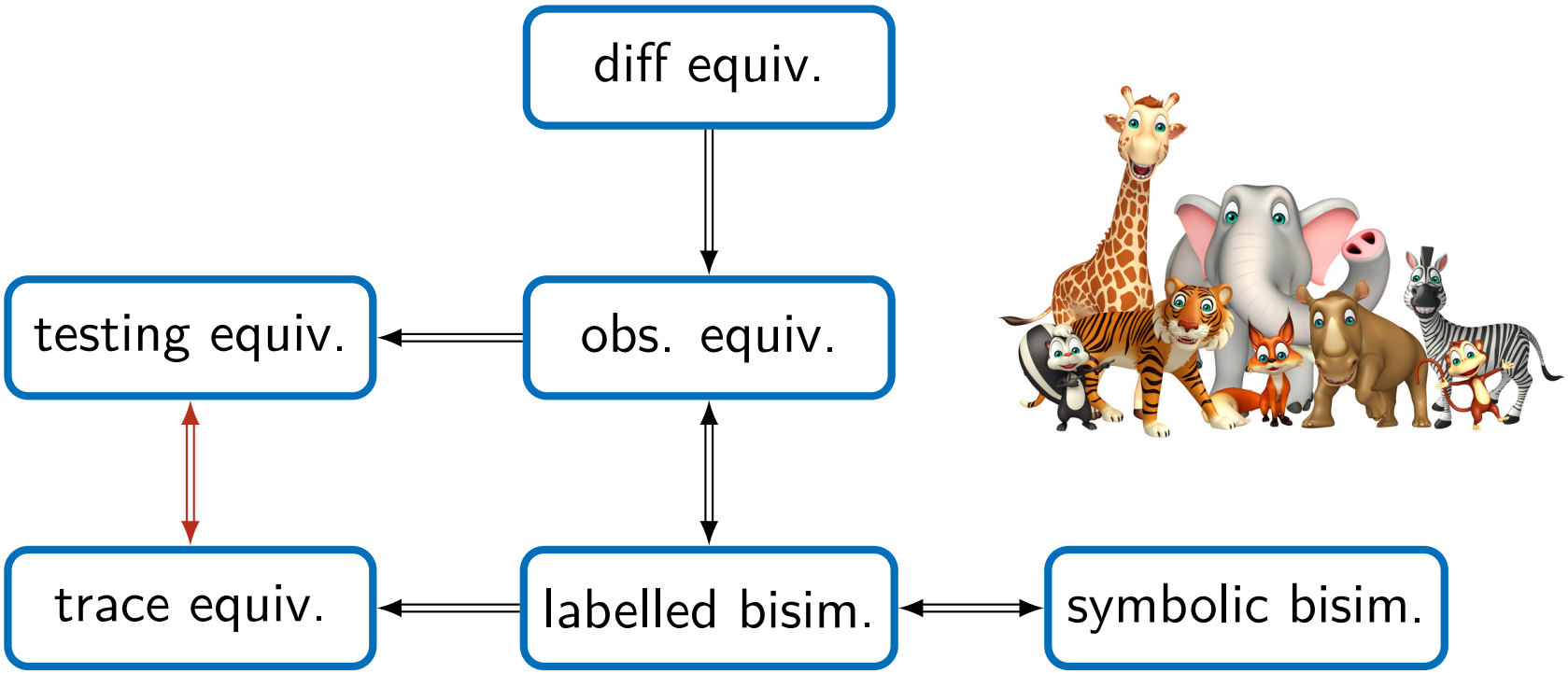
Liu, Lin. A complete symbolic bisimulation for full applied pi calculus.TCS'12

A tour to the (equivalence) zoo



Cheval et al.: Deciding equivalence-based properties using constraint solving. TCS'13

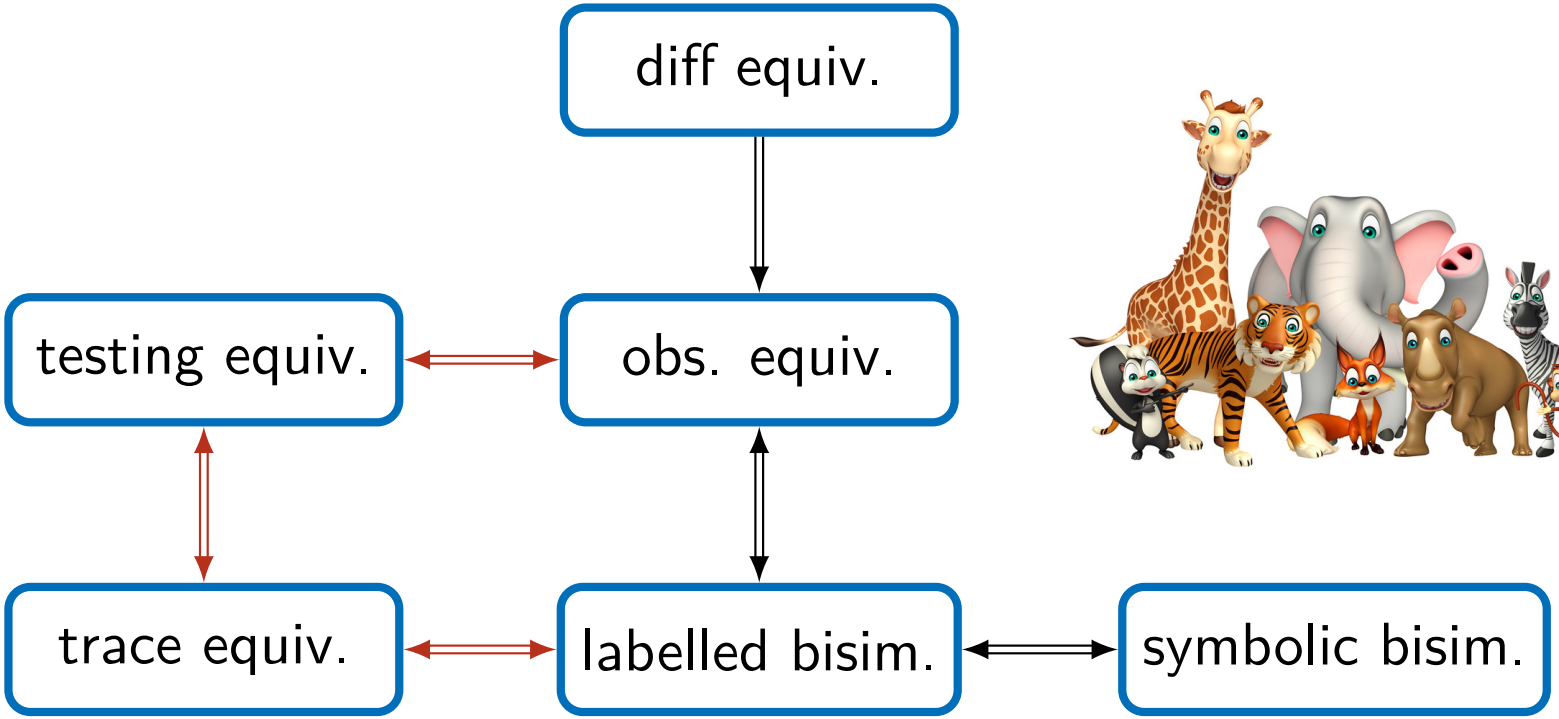
A tour to the (equivalence) zoo



For a **bounded number of sessions** (no replication).

Cheval et al.: Deciding equivalence-based properties using constraint solving. TCS'13

A tour to the (equivalence) zoo



For a class of **determinate processes**.

Cheval et al.: Deciding equivalence-based properties using constraint solving. TCS'13

A few security properties

"Strong" secrecy (non-interference)

$$\mathsf{in}(c, x_1).\mathsf{in}(c, x_2).P\{^{x_1}/_{s}\} \approx \mathsf{in}(c, x_1).\mathsf{in}(c, x_2).P\{^{x_2}/_{s}\}$$

A few security properties

"Strong" secrecy (non-interference)

$$\mathsf{in}(c, x_1).\mathsf{in}(c, x_2).P\{^{x_1}/_{s}\} \approx \mathsf{in}(c, x_1).\mathsf{in}(c, x_2).P\{^{x_2}/_{s}\}$$

Real-or-random secrecy

$$P.\mathsf{out}(c, s) \approx P.\mathsf{new}\ r.\mathsf{out}(c, r)$$

A few security properties

"Strong" secrecy (non-interference)

$$\mathsf{in}(c, x_1).\mathsf{in}(c, x_2).P\{^{x_1}/_{s}\} \approx \mathsf{in}(c, x_1).\mathsf{in}(c, x_2).P\{^{x_2}/_{s}\}$$

Real-or-random secrecy

$$P.\mathsf{out}(c, s) \approx P.\mathsf{new}\ r.\mathsf{out}(c, r)$$

Simulation based security (I is an ideal functionality)

$$\exists S.\ P \approx S[I]$$

A few security properties

"Strong" secrecy (non-interference)

$$\mathsf{in}(c, x_1).\mathsf{in}(c, x_2).P\{^{x_1}/_{s}\} \approx \mathsf{in}(c, x_1).\mathsf{in}(c, x_2).P\{^{x_2}/_{s}\}$$

Real-or-random secrecy

$$P.\mathsf{out}(c, s) \approx P.\mathsf{new}\ r.\mathsf{out}(c, r)$$

Simulation based security (I is an ideal functionality)

$$\exists S.\ P \approx S[I]$$

Anonymity

$$P\{^{a}/_{\mathsf{id}}\} \approx P\{^{b}/_{\mathsf{id}}\}$$

A few security properties

"Strong" secrecy (non-interference)

$$\mathsf{in}(c, x_1).\mathsf{in}(c, x_2).P\{^{x_1}/_{s}\} \approx \mathsf{in}(c, x_1).\mathsf{in}(c, x_2).P\{^{x_2}/_{s}\}$$

Real-or-random secrecy

$$P.\mathsf{out}(c, s) \approx P.\mathsf{new}\ r.\mathsf{out}(c, r)$$

Simulation based security (I is an ideal functionality)

$$\exists S.\ P \approx S[I]$$

Anonymity

$$P\{^{a}/_{\mathsf{id}}\} \approx P\{^{b}/_{\mathsf{id}}\}$$

Vote privacy

Unlinkability

How to model vote privacy?

How can we model

"**the attacker does not learn my vote (0 or 1)**"?

How to model vote privacy?

How can we model

"**the attacker does not learn my vote (0 or 1)**"?

- The attacker cannot learn the value of my vote

How to model vote privacy?

How can we model

"**the attacker does not learn my vote (0 or 1)**"?

- The attacker cannot learn the value of my vote
  $\rightsquigarrow$ but the attacker knows values 0 and 1

How to model vote privacy?

How can we model

"**the attacker does not learn my vote (0 or 1)**"?

- ~~The attacker cannot learn the value of my vote~~
- The attacker cannot distinguish A votes and B votes: $V_A(v) \approx V_B(v)$

How to model vote privacy?

How can we model

"**the attacker does not learn my vote (0 or 1)**"?

- ~~The attacker cannot learn the value of my vote~~

- The attacker cannot distinguish A votes and B votes:
  $V_A(v) \approx V_B(v)$
  $\rightsquigarrow$ but identities are revealed

How to model vote privacy?

How can we model

"**the attacker does not learn my vote (0 or 1)**"?

- ~~The attacker cannot learn the value of my vote~~
- ~~The attacker cannot distinguish A votes and B votes: $V_A(v) \approx V_B(v)$~~
- The attacker cannot distinguish A votes 0 and A votes 1: $V_A(0) \approx V_A(1)$

How to model vote privacy?

How can we model

"**the attacker does not learn my vote (0 or 1)**"?

- ~~The attacker cannot learn the value of my vote~~
- ~~The attacker cannot distinguish A votes and B votes: $V_A(v) \approx V_B(v)$~~
- The attacker cannot distinguish A votes 0 and A votes 1: $V_A(0) \approx V_A(1)$
  $\rightsquigarrow$ but election outcome is revealed

How to model vote privacy?

How can we model

"**the attacker does not learn my vote (0 or 1)**"?

- ~~The attacker cannot learn the value of my vote~~
- ~~The attacker cannot distinguish A votes and B votes: $V_A(v) \approx V_B(v)$~~
- ~~The attacker cannot distinguish A votes 0 and A votes 1: $V_A(0) \approx V_A(1)$~~
- The attacker cannot distinguish the situation where two honest voters swap votes:

$$V_A(0) \parallel V_B(1) \approx V_A(1) \parallel V_B(0)$$

Kremer, Ryan: Analysis of an E-Voting Protocol in the Applied Pi Calculus. ESOP'05

How to verify vote privacy?

Definitions of privacy and stronger variants (receipt-freeness and coercion-resistance) in terms of **process equivalences**.

Our first case study: the **FOO protocol** based on blind signatures

Kremer, Ryan: Analysis of an E-Voting Protocol in the Applied Pi Calculus. ESOP'05

Delaune et al.: Coercion-Resistance and Receipt-Freeness in E-Voting. CSFW'06

How to verify vote privacy?

Definitions of privacy and stronger variants (receipt-freeness and coercion-resistance) in terms of **process equivalences**.

Our first case study: the **FOO protocol** based on blind signatures

- ProVerif was the only tool able to check equivalence properties
- Diff-equivalence checked by ProVerif is too fine-grained
- Needed to do hand proofs

How to verify vote privacy?

Definitions of privacy and stronger variants (receipt-freeness and coercion-resistance) in terms of **process equivalences**.

Our first case study: the **FOO protocol** based on blind signatures

- ProVerif was the only tool able to check equivalence properties
- Diff-equivalence checked by ProVerif is too fine-grained
- Needed to do hand proofs

$\rightsquigarrow$ Motivation for an **alternate tool**.

How to verify vote privacy?

Definitions of privacy and stronger variants (receipt-freeness and coercion-resistance) in terms of **process equivalences**.

Our first case study: the **FOO protocol** based on blind signatures

- ProVerif was the only tool able to check equivalence properties
- Diff-equivalence checked by ProVerif is too fine-grained
- Needed to do hand proofs

$\rightsquigarrow$ Motivation for an **alternate tool**.

see Ben Smyth's talk in next session

AKiSs: our goals and approach

Decision procedure for **trace equivalence**:

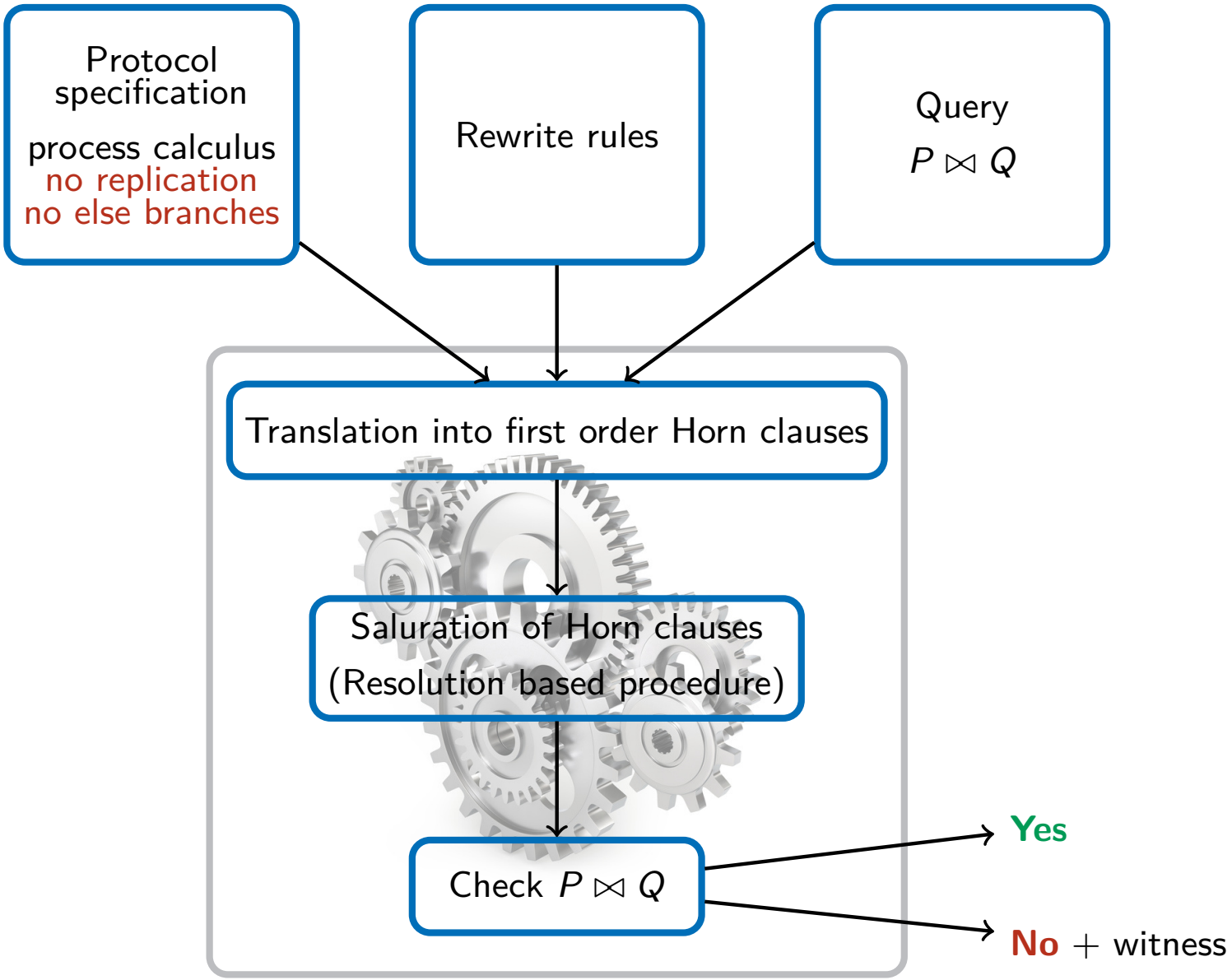
- many equational theories,
- practical implementation

Protocols modelled as **first order Horn clauses** (**bounded number of sessions**, i.e., no replication)

Resolution based procedure for trace equivalence for convergent equational theories (that have the **finite variant property**)

Chadha et al.: Automated Verification of Equivalence Properties of Cryptographic Protocols. ESOP'12, TOCL'16

AKiSs: overview



Modelling protocols in Horn clauses: an example

$$\mathsf{R} = \{dec(enc(x, y), y) \rightarrow x\}$$

$$T = \mathbf{in}(c, x).\text{if } \mathrm{dec}(x, k) = a \text{ then } \mathbf{out}(c, s)$$

$$\begin{aligned}
\mathsf{r}_{\mathbf{in}(c,x)} &\Leftarrow \mathsf{k}(X, x) \\
\mathsf{r}_{\mathbf{in}(c,x),\mathbf{test}} &\Leftarrow \mathsf{k}(X, x), \mathrm{dec}(x, k) =_{\mathsf{R}} a \\
\mathsf{r}_{\mathbf{in}(c,x),\mathbf{test},\mathbf{out}(c)} &\Leftarrow \mathsf{k}(X, x), \mathrm{dec}(x, k) =_{\mathsf{R}} a \\
\\
\mathsf{k}_{\mathbf{in}(c,x),\mathbf{test},\mathbf{out}(c)}(w_1, s) &\Leftarrow \mathsf{k}(X, x), \mathrm{dec}(x, k) =_{\mathsf{R}} a
\end{aligned}$$

Modelling protocols in Horn clauses: an example

$$\mathrm{R} = \{dec(enc(x, y), y) \to x\}$$

$$T = \mathbf{in}(c, x).\text{if } \mathrm{dec}(x, k) = a \text{ then } \mathbf{out}(c, s)$$

$$\begin{aligned}
\mathrm{r}_{\mathbf{in}(c,x)} &\Leftarrow \mathrm{k}(X, x) \\
\mathrm{r}_{\mathbf{in}(c,x),\mathbf{test}} &\Leftarrow \mathrm{k}(X, x), \mathrm{dec}(x, k) =_{\mathrm{R}} a \\
\mathrm{r}_{\mathbf{in}(c,x),\mathbf{test},\mathbf{out}(c)} &\Leftarrow \mathrm{k}(X, x), \mathrm{dec}(x, k) =_{\mathrm{R}} a \\
\\
\mathrm{k}_{\mathbf{in}(c,x),\mathbf{test},\mathbf{out}(c)}(w_1, s) &\Leftarrow \mathrm{k}(X, x), \mathrm{dec}(x, k) =_{\mathrm{R}} a
\end{aligned}$$

Get rid of equalities by **equational unification**.

$$\mathrm{mgu}_{\mathrm{R}}(\mathrm{dec}(x, k) =_{\mathrm{R}} a) : x \mapsto \mathrm{enc}(a, k)$$

Modelling protocols in Horn clauses: an example

$$\mathrm{R} = \{dec(enc(x, y), y) \to x\}$$

$$T = \mathbf{in}(c, x).\text{if } \mathrm{dec}(x, k) = a \text{ then } \mathbf{out}(c, s)$$

$$\begin{aligned}
\mathrm{r}_{\mathbf{in}(c,x)} &\Leftarrow \mathrm{k}(X, x) \\
\mathrm{r}_{\mathbf{in}(c,\mathrm{enc}(a,k)),\mathbf{test}} &\Leftarrow \mathrm{k}(X, \mathrm{enc}(a, k)) \\
\mathrm{r}_{\mathbf{in}(c,\mathrm{enc}(a,k)),\mathbf{test},\mathbf{out}(c)} &\Leftarrow \mathrm{k}(X, \mathrm{enc}(a, k))
\end{aligned}$$

$$\mathrm{k}_{\mathbf{in}(c,\mathrm{enc}(a,k)),\mathbf{test},\mathbf{out}(c)}(w_1, s) \Leftarrow \mathrm{k}(X, \mathrm{enc}(a, k))$$

Get rid of equalities by **equational unification**.

$$\mathrm{mgu}_{\mathrm{R}}(\mathrm{dec}(x, k) =_{\mathrm{R}} a) : x \mapsto \mathrm{enc}(a, k)$$

Modelling protocols in Horn clauses: an example

$$\mathrm{R} = \{dec(enc(x, y), y) \rightarrow x\}$$

$$T = \mathbf{in}(c, x).\text{if } \mathrm{dec}(x, k) = a \text{ then } \mathbf{out}(c, s)$$

$$\begin{aligned} \mathrm{k}(\mathrm{enc}(X, Y), \mathrm{enc}(x, y)) &\Leftarrow \mathrm{k}(X, x), \mathrm{k}(Y, y) \\ \mathrm{k}(\mathrm{dec}(X, Y), \mathrm{dec}(x, y)) &\Leftarrow \mathrm{k}(X, x), \mathrm{k}(Y, y) \end{aligned}$$

Modelling protocols in Horn clauses: an example

$$\mathrm{R} = \{dec(enc(x, y), y) \to x\}$$

$$T = \mathbf{in}(c, x).\text{if } \mathrm{dec}(x, k) = a \text{ then } \mathbf{out}(c, s)$$

$$\begin{aligned} \mathrm{k}(\mathrm{enc}(X, Y), \mathrm{enc}(x, y)) &\Leftarrow \mathrm{k}(X, x), \mathrm{k}(Y, y) \\ \mathrm{k}(\mathrm{dec}(X, Y), \mathrm{dec}(x, y)) &\Leftarrow \mathrm{k}(X, x), \mathrm{k}(Y, y) \end{aligned}$$

Rewrite systems with the **finite variant property**:

precompute all possible normal forms

$\rightsquigarrow$ **get rid of** equational reasoning

Modelling protocols in Horn clauses: an example

$$\mathrm{R} = \{dec(enc(x, y), y) \to x\}$$

$$T = \mathbf{in}(c, x).\text{if } \mathrm{dec}(x, k) = a \text{ then } \mathbf{out}(c, s)$$

$$\begin{aligned} \mathrm{k}(\mathrm{enc}(X, Y), \mathrm{enc}(x, y)) &\Leftarrow \mathrm{k}(X, x), \mathrm{k}(Y, y) \\ \mathrm{k}(\mathrm{dec}(X, Y), \mathrm{dec}(x, y)) &\Leftarrow \mathrm{k}(X, x), \mathrm{k}(Y, y) \\ \mathrm{k}(\mathrm{dec}(X, Y), z) &\Leftarrow \mathrm{k}(X, \mathrm{enc}(z, y)), \mathrm{k}(Y, y) \end{aligned}$$

Rewrite systems with the **finite variant property**:

precompute all possible normal forms

$\rightsquigarrow$ **get rid of** equational reasoning

Saturating clauses

A clause is **solved** if it is of the form

$$H \Leftarrow \mathsf{k}_{w_1}(X_1, x_1), \ldots, \mathsf{k}_{w_n}(X_n, x_n)$$

Resolution

$$\frac{\begin{array}{l} H \Leftarrow \mathsf{k}_{uv}(X, t), B_1, \ldots, B_n \in K, \quad \mathsf{k}_w(R, t') \Leftarrow B_{n+1}, \ldots, B_m \in K_{\text{solved}} \\ t \text{ not a var} \qquad \sigma = \text{mgu}(\mathsf{k}_u(X, t), \mathsf{k}_w(R, t')) \end{array}}{K := K \cup \Big((H \Leftarrow B_1, \ldots, B_m)\sigma \Big)}$$

Identity

$$\frac{\begin{array}{l} \mathsf{k}_u(R, t) \Leftarrow B_1, \ldots, B_n \in K_{\text{solved}} \quad \mathsf{k}_{u'v'}(R', t') \Leftarrow B_{n+1}, \ldots, B_m \in K_{\text{solved}} \\ \sigma = \text{mgu}(\mathsf{k}_u(\_, t), \mathsf{k}_{u'}(\_, t')) \end{array}}{K = K \cup \Big((\mathsf{i}_{u'v'}(R, R') \Leftarrow B_1, \ldots, B_m)\sigma \Big)}$$

Iterated until reaching fixpoint.

Properties of saturated set of clauses

A the end of the saturation we have a **finite set of solved clauses** that represents:

- all reachable traces of the protocol
- all deducible messages by the adversary
- all identities among adversary recipes

Trace equivalences

Trace equivalence: $P \sqsubseteq_t Q$

if $(P, \emptyset) \stackrel{\text{tr}}{\Rightarrow} (P', \varphi)$ then $\exists Q', \varphi'.\ (Q, \emptyset) \stackrel{\text{tr}}{\Rightarrow} (Q', \varphi') \wedge \varphi \sim_s \varphi'$

$P \approx Q$ iff $P \sqsubseteq Q \wedge Q \sqsubseteq P$

Trace equivalences

Fine grained trace equivalence: $P \sqsubseteq_{ft} Q$

$\forall$ interleaving T of P. $\exists$ interleaving T' of Q. $T \approx_t T'$

Trace equivalence: $P \sqsubseteq_t Q$

if $(P, \emptyset) \stackrel{\text{tr}}{\Rightarrow} (P', \varphi)$ then $\exists Q', \varphi'. (Q, \emptyset) \stackrel{\text{tr}}{\Rightarrow} (Q', \varphi') \wedge \varphi \sim_s \varphi'$

$P \approx Q$ iff $P \sqsubseteq Q \wedge Q \sqsubseteq P$

Trace equivalences

Fine grained trace equivalence: $P \sqsubseteq_{ft} Q$

$\forall$ interleaving T of P. $\exists$ interleaving T' of Q. $T \approx_t T'$

Trace equivalence: $P \sqsubseteq_t Q$

if $(P, \emptyset) \stackrel{\text{tr}}{\Rightarrow} (P', \varphi)$ then $\exists Q', \varphi'.\ (Q, \emptyset) \stackrel{\text{tr}}{\Rightarrow} (Q', \varphi') \wedge \varphi \sim_s \varphi'$

$P \approx Q$ iff $P \sqsubseteq Q \wedge Q \sqsubseteq P$

Trace equivalences

Fine grained trace equivalence: $P \sqsubseteq_{ft} Q$

$\forall$ interleaving T of P. $\exists$ interleaving T' of Q. $T \approx_t T'$

Trace equivalence: $P \sqsubseteq_t Q$

if $(P, \emptyset) \stackrel{\text{tr}}{\Rightarrow} (P', \varphi)$ then $\exists Q', \varphi'.\ (Q, \emptyset) \stackrel{\text{tr}}{\Rightarrow} (Q', \varphi') \wedge \varphi \sim_s \varphi'$

Coarse trace equivalence: $P \sqsubseteq_{ct} Q$

if $(P, \emptyset) \stackrel{\text{tr}}{\Rightarrow} (P', \varphi) \wedge (r = s)\varphi$ then $\exists Q', \varphi'.\ (Q, \emptyset) \stackrel{\text{tr}}{\Rightarrow} (Q', \varphi') \wedge (r = s)\varphi'$

$P \approx Q$ iff $P \sqsubseteq Q \wedge Q \sqsubseteq P$

Trace equivalences

Fine grained trace equivalence: $P \sqsubseteq_{ft} Q$

$\forall$ interleaving T of P. $\exists$ interleaving T' of Q. $T \approx_t T'$

$\not\cap$

Trace equivalence: $P \sqsubseteq_t Q$

if $(P, \emptyset) \stackrel{\text{tr}}{\Rightarrow} (P', \varphi)$ then $\exists Q', \varphi'.\ (Q, \emptyset) \stackrel{\text{tr}}{\Rightarrow} (Q', \varphi') \wedge \varphi \sim_s \varphi'$

$\not\cap$

Coarse trace equivalence: $P \sqsubseteq_{ct} Q$

if $(P, \emptyset) \stackrel{\text{tr}}{\Rightarrow} (P', \varphi) \wedge (r = s)\varphi$ then $\exists Q', \varphi'.\ (Q, \emptyset) \stackrel{\text{tr}}{\Rightarrow} (Q', \varphi') \wedge (r = s)\varphi'$

$P \approx Q$ iff $P \sqsubseteq Q \wedge Q \sqsubseteq P$

Trace equivalences

Fine grained trace equivalence: $P \sqsubseteq_{ft} Q$

$\forall$ interleaving T of P. $\exists$ interleaving T' of Q. $T \approx_t T'$

Trace equivalence: $P \sqsubseteq_t Q$

if $(P, \emptyset) \stackrel{\text{tr}}{\Rightarrow} (P', \varphi)$ then $\exists Q', \varphi'.\ (Q, \emptyset) \stackrel{\text{tr}}{\Rightarrow} (Q', \varphi') \wedge \varphi \sim_s \varphi'$

det.
proc.

Coarse trace equivalence: $P \sqsubseteq_{ct} Q$

if $(P, \emptyset) \stackrel{\text{tr}}{\Rightarrow} (P', \varphi) \wedge (r = s)\varphi$ then $\exists Q', \varphi'.\ (Q, \emptyset) \stackrel{\text{tr}}{\Rightarrow} (Q', \varphi') \wedge (r = s)\varphi'$

$P \approx Q$ iff $P \sqsubseteq Q \wedge Q \sqsubseteq P$

P is *determinate* if whenever $(P, \emptyset) \stackrel{\text{tr}}{\Rightarrow} (T, \varphi)$ and $(P, \emptyset) \stackrel{\text{tr}}{\Rightarrow} (T', \varphi')$ then $\varphi \sim_s \varphi'$.

AKiSs: checking equivalences

AKiSs can be used to

- **under-approximate** trace equivalence : prove $\approx_{ft}$
- **over-approximate** trace equivalence : prove $\not\approx_{ct}$
- **prove trace equivalence** for determinate processes

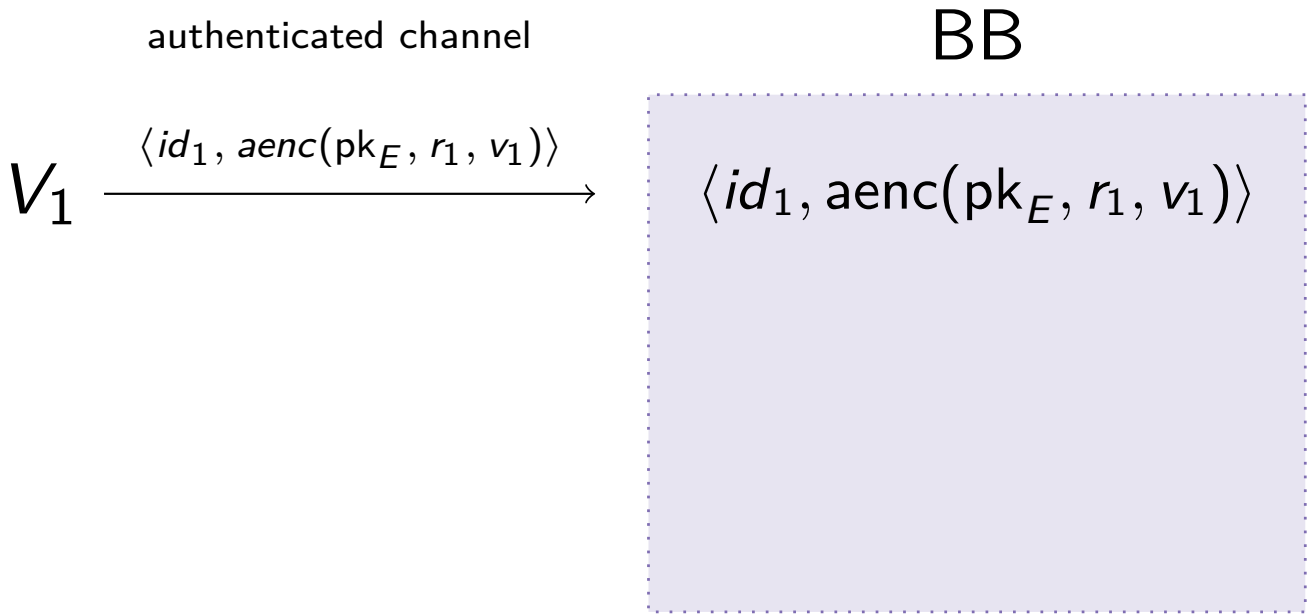
Correctness:
any convergent rewrite system that has the finite variant property
no else branches

Termination:
guaranteed for any subterm convergent rewrite system
$\ell \to r$: r is either a subterm of ℓ or ground

Terminates in practice on other examples as well

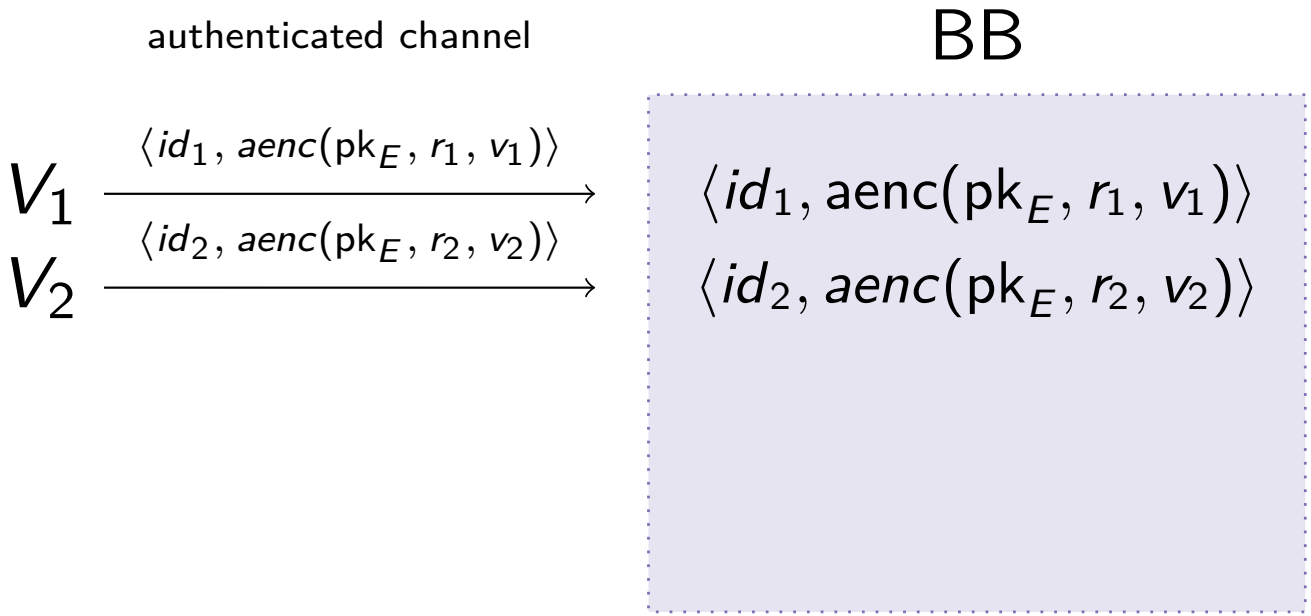
First automated proof of FOO e-voting protocol

The Helios e-voting protocol (MixNet version)



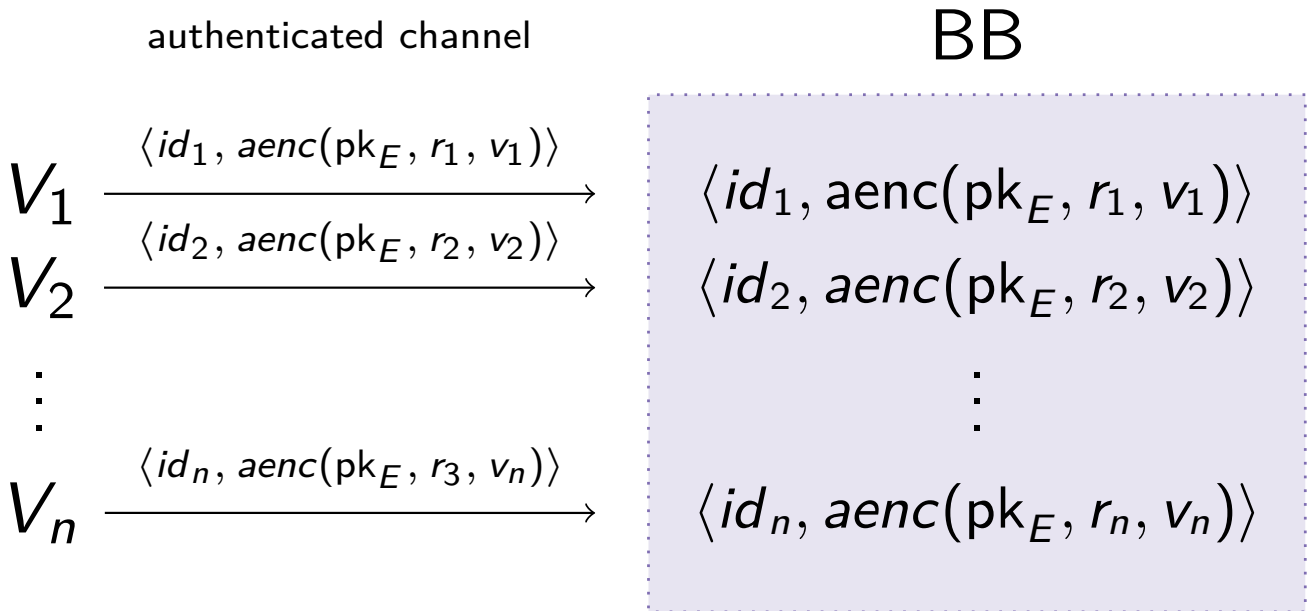
where pk_E is the election public key and MIX a verifiable mixnet.

The Helios e-voting protocol (MixNet version)



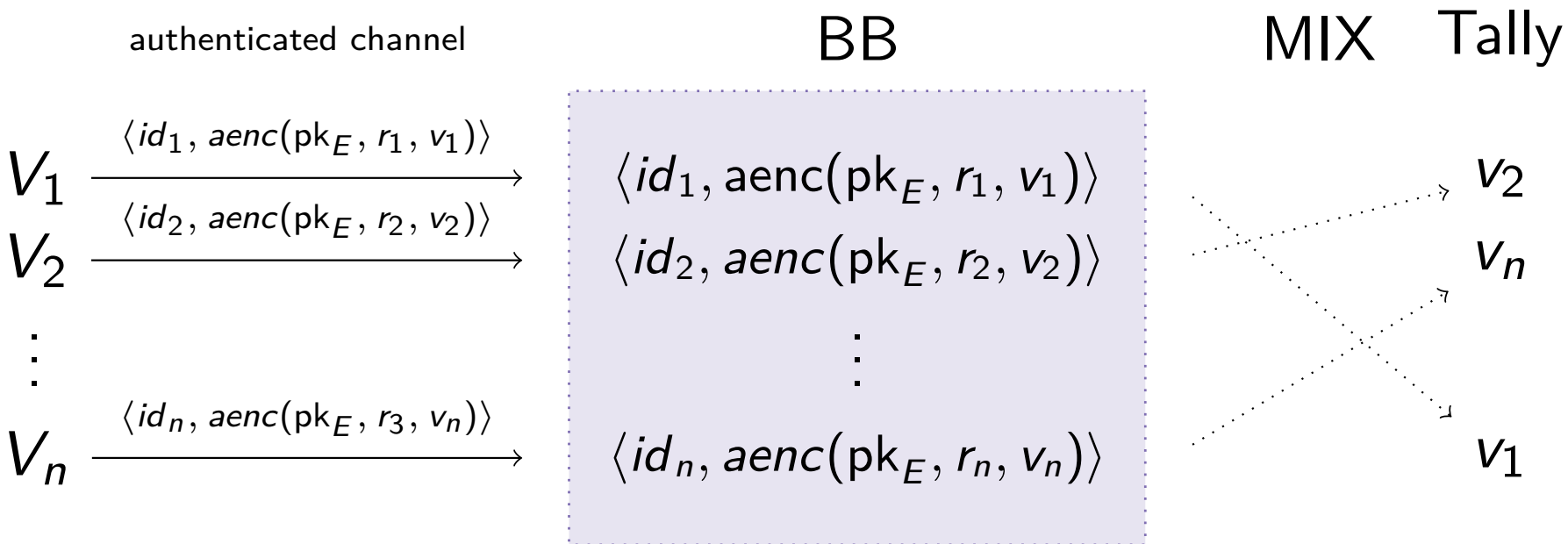
where pk_E is the election public key and MIX a verifiable mixnet.

The Helios e-voting protocol (MixNet version)



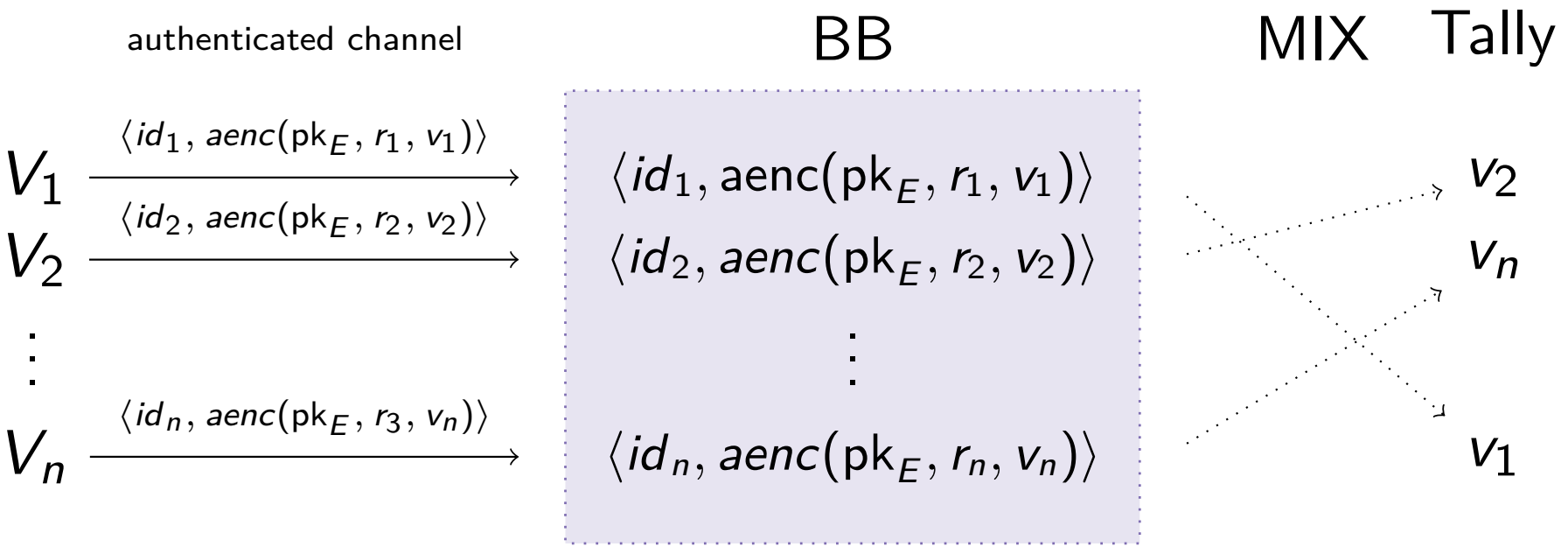
where pk_E is the election public key and MIX a verifiable mixnet.

The Helios e-voting protocol (MixNet version)



where pk_E is the election public key and MIX a verifiable mixnet.

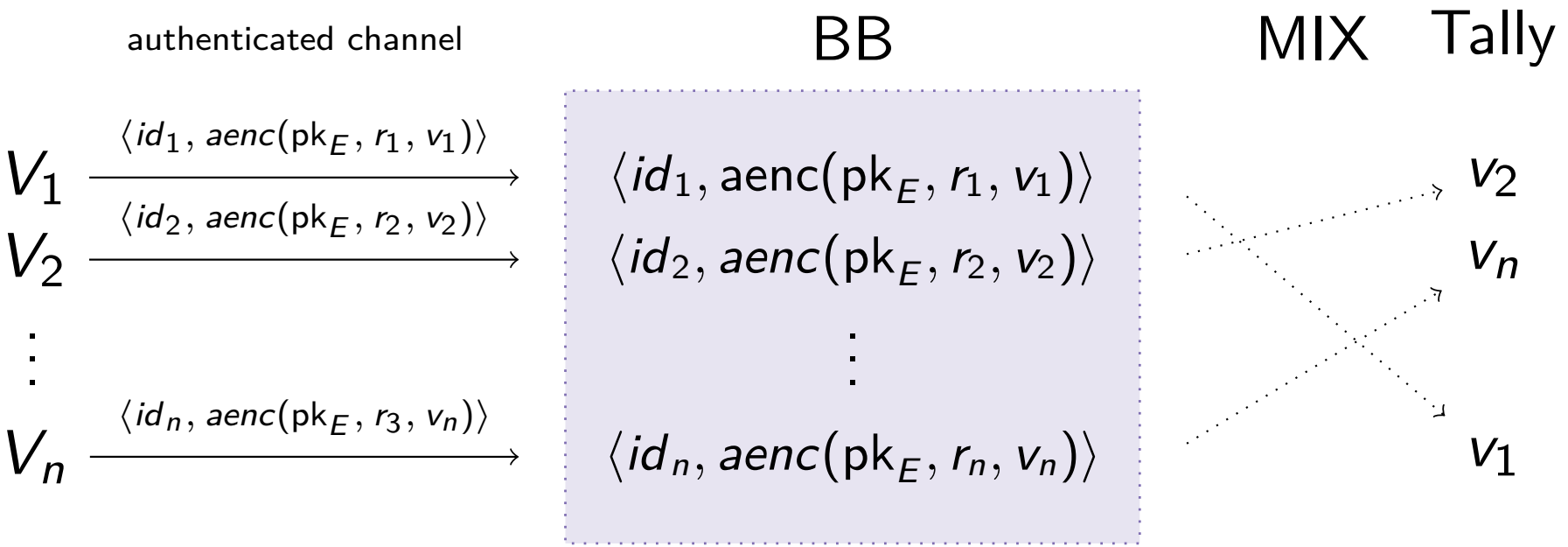
The Helios e-voting protocol (MixNet version)



where pk_E is the election public key and MIX a verifiable mixnet.

Privacy: $\mathrm{Helios}(v_1, v_2) \stackrel{?}{\approx}_t \mathrm{Helios}(v_2, v_1)$

The Helios e-voting protocol (MixNet version)

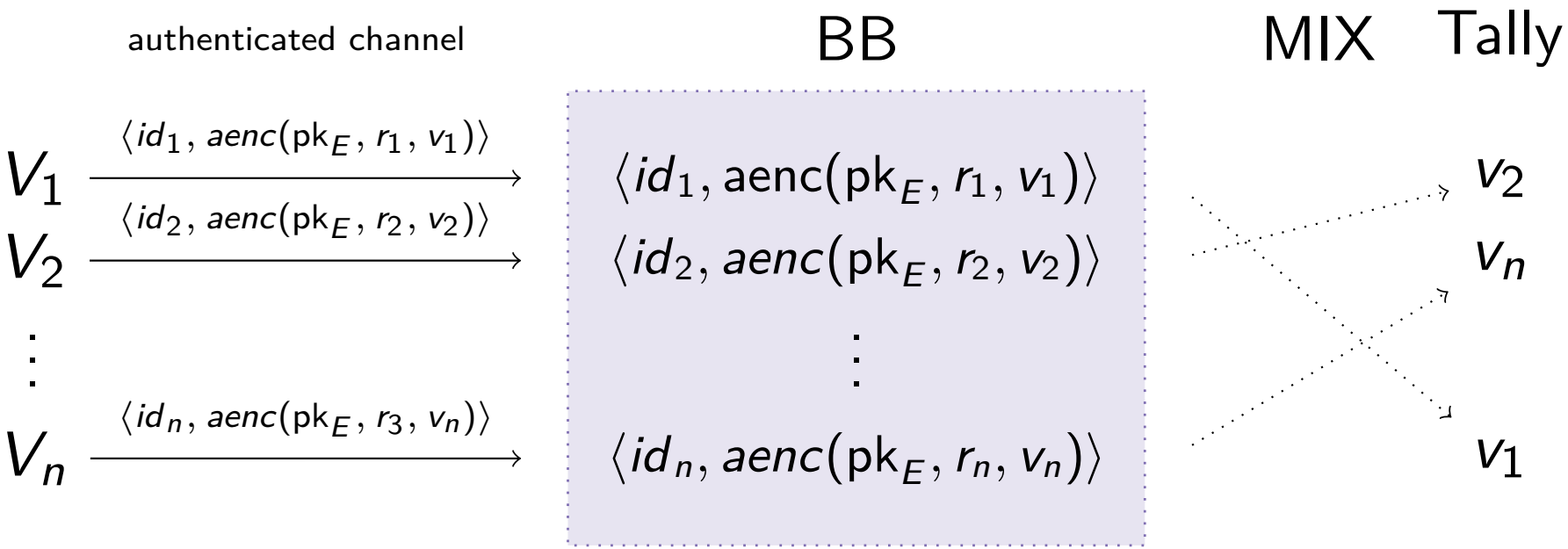


where pk_E is the election public key and MIX a verifiable mixnet.

Privacy: $\mathrm{Helios}(v_1, v_2) \stackrel{?}{\approx}_t \mathrm{Helios}(v_2, v_1) \rightsquigarrow$ **replay attack!**

Cortier,Smyth: Attacking and Fixing Helios: An Analysis of Ballot Secrecy. CSF'11

The Helios e-voting protocol (MixNet version)



where pk_E is the election public key and MIX a verifiable mixnet.

Privacy: $\mathrm{Helios}(v_1, v_2) \overset{?}{\approx}_t \mathrm{Helios}(v_2, v_1)$ $\rightsquigarrow$ **replay attack!**

Fix: either use weeding, or zkp that voter knows encryption randomness

Everlasting privacy

Does verifiability decrease vote privacy?

Publishing encrypted votes on the bulletin board may be **a threat for vote privacy**.

- Future technology and scientific advances may break encryptions
- How long must a vote remain private?
  1 year? 10 years? 100 years? 10^{10} years?
- Impossible to predict the necessary key length with certainty:
  typical recommendations for less than 10 years
  (cf www.keylength.com)

Everlasting privacy

Does verifiability decrease vote privacy?

Publishing encrypted votes on the bulletin board may be **a threat for vote privacy**.

- Future technology and scientific advances may break encryptions
- How long must a vote remain private?
  1 year? 10 years? 100 years? 10^{10} years?
- Impossible to predict the necessary key length with certainty:
  typical recommendations for less than 10 years
  (cf www.keylength.com)

$\rightsquigarrow$ **everlasting privacy**: guarantee privacy even if crypto is broken

Modelling everlasting privacy

- Information available in the future: **everlasting channels**
- Define future attacker capabilities (crypto assumption broken)
  $\rightsquigarrow$ equational theory E^+
  Example: $\mathsf{break}(\mathsf{aenc}(\mathsf{pk}(x), y, z)) \rightarrow z$
- Check in **two phases**:
  1. check trace equivalence with E
  2. check static equivalence with E^+ on future information

$\rightsquigarrow$ **implemented in AKiSs and ProVerif**

Arapinis et al.: Practical Everlasting Privacy. POST'13

Modelling everlasting privacy

- Information available in the future: **everlasting channels**
- Define future attacker capabilities (crypto assumption broken)
  $\rightsquigarrow$ equational theory E^+
  Example: $\mathsf{break}(\mathsf{aenc}(\mathsf{pk}(x), y, z)) \rightarrow z$
- Check in **two phases**:
  1. check trace equivalence with E
  2. check static equivalence with E^+ on future information

$\rightsquigarrow$ **implemented in AKiSs and ProVerif**

Achieving everlasting privacy:

- Do not publish encryption on the BB, but only a perfectly hiding commitment
- Replace identities by anonymous credentials $\rightsquigarrow$ Belenios

Arapinis et al.: Practical Everlasting Privacy. POST'13

How to model unlinkability

Unlinkability [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.*

Applications: e-Passport, mobile phones, RFID tags, ...

How to model unlinkability

Unlinkability [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.*

Applications: e-Passport, mobile phones, RFID tags, ...

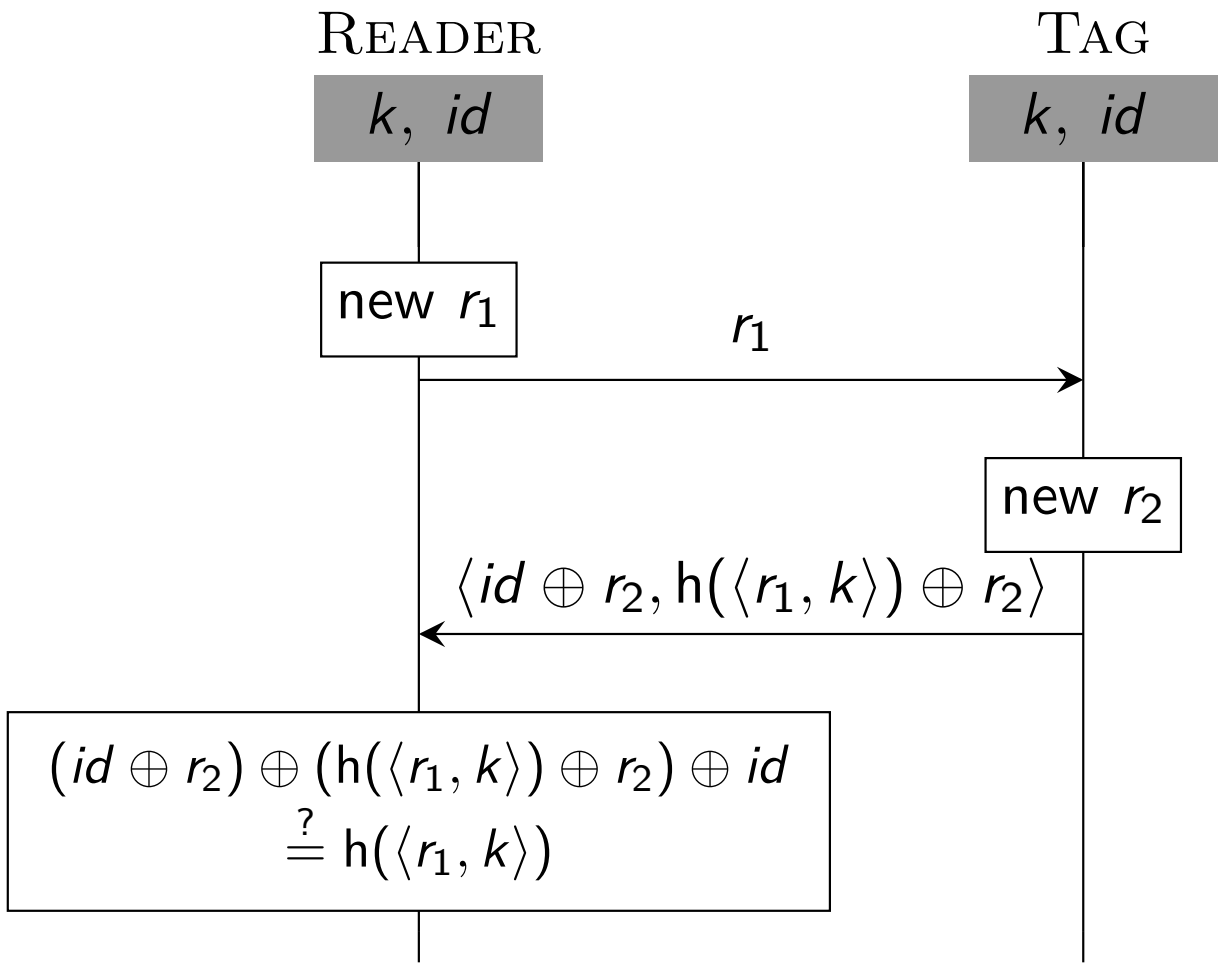
Can be modelled as an equivalence property:

2 sessions of the **same** device $\approx$ 2 sessions of **different** devices

Arapinis et al. Analysing Unlinkability and Anonymity Using the Applied Pi Calculus. CSF'10

Brusò et al. Formal Verification of Privacy for RFID Systems. CSF'10

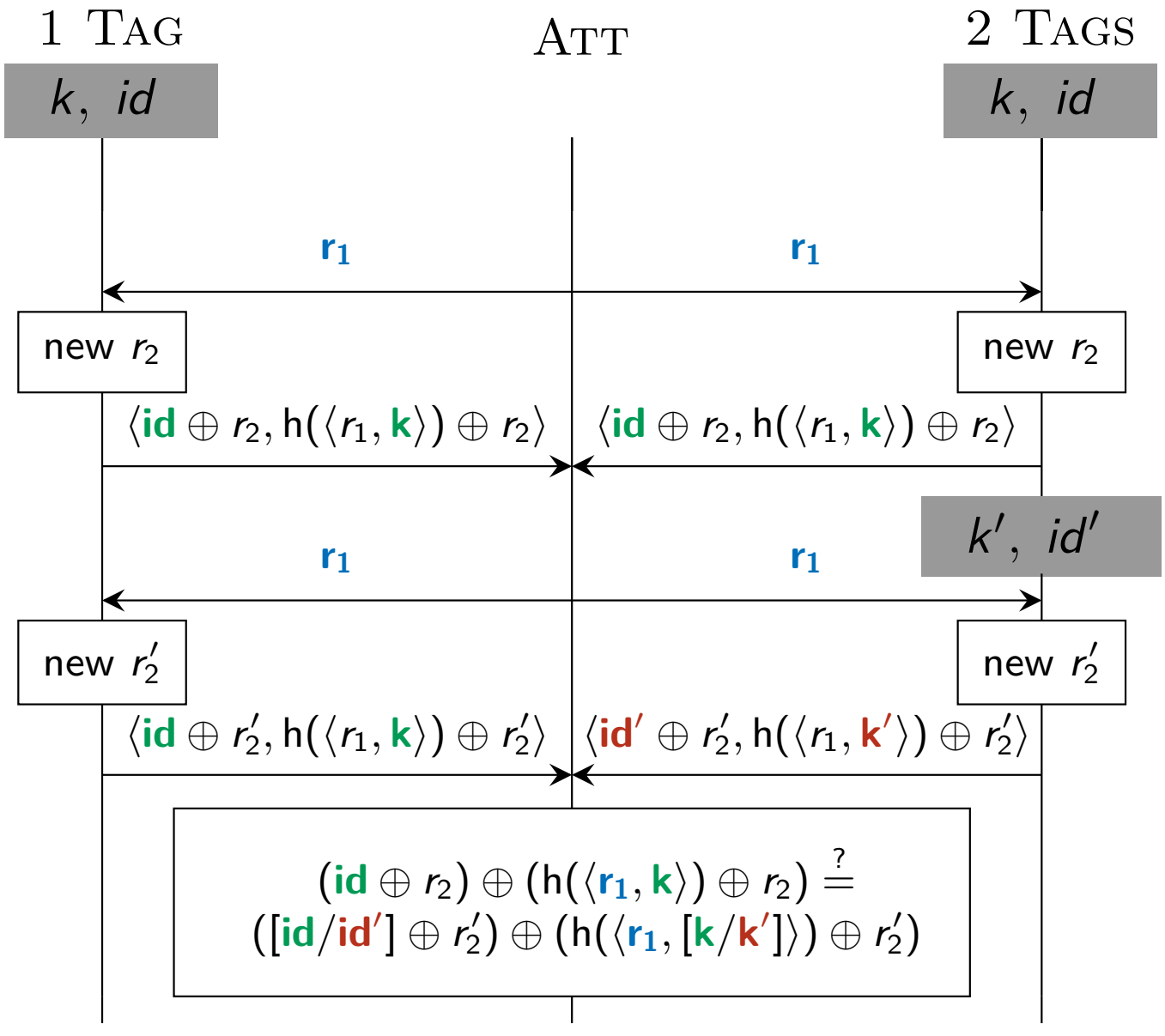
Authentication protocol of a RFID tag (KCL)



Is unlinkability satisfied?

$$\mathrm{tag}(id, k) \mid \mathrm{tag}(id, k) \stackrel{?}{\approx} \mathrm{tag}(id, k) \mid \mathrm{tag}(id', k')$$

Linkability attack



Automated analysis of KCL?

Which tool to choose?

Automated analysis of KCL?

Which tool to choose?

- None provides **support for** $\oplus$

Automated analysis of KCL?

Which tool to choose?

- None provides **support for** $\oplus$
- Abstracting away from **algebraic properties**: we miss the linkability attack

Automated analysis of KCL?

Which tool to choose?

- None provides **support for** $\oplus$
- Abstracting away from **algebraic properties**: we miss the linkability attack

Motivated an **extension of AKiSs with** $\oplus$:

joint work with Baelde, Delaune and Gazeau

- perform **Horn clause resolution modulo AC**
- new **strategy**: forbid some resolutions to avoid non-termination
  $\rightsquigarrow$ major changes in the completeness proof
- successfully tested among others on **5 RFID protocols**

Overview of tools

Unbounded number of sessions (no termination guarantees)

	ProVerif	**Tamarin**	**Maude NPA**
equivalence	diff (+ extensions)	diff	diff
protocol model	applied pi	MSR (state, else, ...)	strands (no else)
eq. theories	finite variant (?)	subterm conv. + DH	finite variant + algebraic prop.

Overview of tools

Unbounded number of sessions (no termination guarantees)

	ProVerif	Tamarin	Maude NPA
equivalence	diff (+ extensions)	diff	diff
protocol model	applied pi	MSR (state, else, ...)	strands (no else)
eq. theories	finite variant (?)	subterm conv. + DH	finite variant + algebraic prop.

Bounded number of sessions

	SPEC	APTE	AKiSs
equivalence	symb. bisimulations	trace equiv	$\sim$ trace equiv
protocol model	spi (no else)	applied pi	applied pi (no else)
eq. theories	fixed	fixed	finite variant + xor

Overview of tools

Unbounded number of sessions (no termination guarantees)

	ProVerif	Tamarin	Maude NPA
equivalence	diff (+ extensions)	diff	diff
protocol model	applied pi	MSR (state, else, ...)	strands (no else)
eq. theories	finite variant (?)	subterm conv. + DH	finite variant + algebraic prop.

Bounded number of sessions

	SPEC	APTE	AKiSs
equivalence	symb. bisimulations	trace equiv	$\sim$ trace equiv
protocol model	spi (no else)	applied pi	applied pi (no else)
eq. theories	fixed	fixed	finite variant + xor

No swiss knife for equivalence properties

Theory and practice of equivalence properties

Extensions of **AKiSs**

- else branches, needed e.g. for analysing unlinkability for the e-Passport
- more algebraic properties, e.g., DH exponentiation à la tamarin

Theory and practice of equivalence properties

Extensions of **AKiSs**

- else branches, needed e.g. for analysing unlinkability for the e-Passport
- more algebraic properties, e.g., DH exponentiation à la tamarin

Merge **APTE and AKISS**

joint work with Cheval

- decide trace equivalence
- general processes (else branches, not necessarily determinate)
- many equational theories

Theory and practice of equivalence properties (2)

Decidability and complexity joint work with Cheval and Rakotonirina

e.g. for subterm convergent equational theories, obs. equivalence is coNP complete for determinate processes, but

coNEXP hard otherwise

$\rightsquigarrow$ interesting insights on how to make tools efficient

see Itsaka's 5 minute talk

Automated Security Proofs of Cryptographic Protocols

- Theory and practice for equivalence properties
- Models for and analysis of secure elements (TPM, SGX, . . .)
- Multi-factor authentication
- E-voting on untrusted clients

Join us: open PhD and post-doc positions