Models and Techniques for Symbolic Analysis of Security Protocols

Episode II: Equivalence properties

Steve Kremer

Summer School : Models and Tools for Cryptographic Proofs

Episode I

Part 1 Protocols

Part 2 Model : the applied-pi calculus
$\rightarrow$ the ProVerif tool

Part 3 Analysis : protocols as Horn clauses

Episode II

Part 4 Indistinguishability properties in the applied pi calculus

Part 5 Applications: modelling security protocols

Part 6 Automated analysis : ProVerif & DEEPSEC

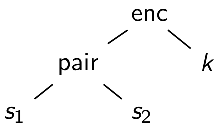
Part I

Indistinguishability properties in the applied pi calculus

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

Modelling the protocol

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

$$
\begin{array}{rcll}
P ::= & & 0 & \\
& \mid & c(x).P & \text{input} \\
& \mid & \overline{c}\langle t\rangle.P & \text{output} \\
& \mid & \text{if } t_1 = t_2 \text{ then } P \text{ else } Q & \text{conditional} \\
& \mid & P \parallel Q & \text{parallel} \\
& \mid & !P & \text{replication} \\
& \mid & \text{new } n.P & \text{restriction}
\end{array}
$$

Modelling the protocol

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

$$
\begin{array}{rcll}
P ::= & & 0 & \\
& | & c(x).P & \text{input} \\
& | & \overline{c}\langle t\rangle.P & \text{output} \\
& | & \text{if } t_1 = t_2 \text{ then } P \text{ else } Q & \text{conditional} \\
& | & P \parallel Q & \text{parallel} \\
& | & !P & \text{replication} \\
& | & \text{new } n.P & \text{restriction}
\end{array}
$$

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**:

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

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

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

$$\varphi \vdash^{\text{dec}(x_1,x_3)} n_1 \qquad \varphi \not\vdash n_2 \qquad \varphi \vdash^{\mathbf{1}} \mathbf{1}$$

Deduction may not be sufficient!

Some properties not captured by the terms an attacker can deduce.

Example

Consider 2 observations by an attacker

$$\varphi_1 = \{{}^{a}/_{x_1}, {}^{0}/_{x_2}, {}^{1}/_{x_3}, {}^{\langle a,0\rangle}/_{x_4}\}$$

$$\varphi_2 = \{{}^{a}/_{x_1}, {}^{0}/_{x_2}, {}^{1}/_{x_3}, {}^{\langle a,1\rangle}/_{x_4}\}$$

Note: set of terms deducible from both frames are identical.

Deduction may not be sufficient!

Some properties not captured by the terms an attacker can deduce.

Example

Consider 2 observations by an attacker

$$\varphi_1 = \{{}^{a}/_{x_1}, {}^{0}/_{x_2}, {}^{1}/_{x_3}, {}^{\langle a,0\rangle}/_{x_4}\}$$

$$\varphi_2 = \{{}^{a}/_{x_1}, {}^{0}/_{x_2}, {}^{1}/_{x_3}, {}^{\langle a,1\rangle}/_{x_4}\}$$

Note: set of terms deducible from both frames are identical.

But the attacker may learn the **link** between a and either 0 or 1.

Deduction may not be sufficient!

Some properties not captured by the terms an attacker can deduce.

Example

Consider 2 observations by an attacker

$$\varphi_1 = \{^{a}/_{x_1}, ^{0}/_{x_2}, ^{1}/_{x_3}, ^{\langle a,0\rangle}/_{x_4}\}$$

$$\varphi_2 = \{^{a}/_{x_1}, ^{0}/_{x_2}, ^{1}/_{x_3}, ^{\langle a,1\rangle}/_{x_4}\}$$

Note: set of terms deducible from both frames are identical.

But the attacker may learn the **link** between a and either 0 or 1.

Such properties are captured by the notion of **indistinguishability**: an attacker is unable to distinguish two frames.

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)

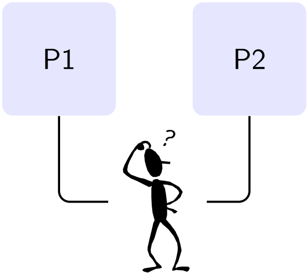
  If B ended a session with A (and parameters p) then A must have started a session with B (and parameters p').

Not all properties can be expressed on a trace.

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

Indistinguishability (informally)

Can the adversary **distinguish two situations**, i.e. decide whether it is interacting with protocol P1 or protocol P2?



Distinguishing messages

The notion of indistinguishability of message sequences is formalised by **static equivalence** of frames.

Idea: any test an attacker can perform on one frame should also hold in the other frame.

Definition (static equivalence)

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

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

Static equivalence: examples

Example

$$\varphi_1 = \{^0/_x, ^1/_y\} \text{ and } \varphi_2 = \{^1/_x, ^0/_y\}$$

Static equivalence: examples

Example

$$\varphi_1 = \{^0/_x, ^1/_y\} \text{ and } \varphi_2 = \{^1/_x, ^0/_y\}$$

$$\varphi_1 \not\sim_s \varphi_2 \text{ as } (x = 0)\varphi_1 \text{ while } (x \neq 0)\varphi_2.$$

Static equivalence: examples

Example

$$\varphi_1 = \{^0/_x, ^1/_y\} \text{ and } \varphi_2 = \{^1/_x, ^0/_y\}$$

$$\varphi_1 \not\sim_s \varphi_2 \text{ as } (x = 0)\varphi_1 \text{ while } (x \neq 0)\varphi_2.$$

Example

$\varphi_1 = \nu k\{^{\mathsf{aenc}(0,\mathsf{pk}(k))}/_x, ^{\mathsf{pk}(k)}/_y\} \quad \varphi_2 = \nu k\{^{\mathsf{aenc}(1,\mathsf{pk}(k))}/_x, ^{\mathsf{pk}(k)}/_y\}$

Static equivalence: examples

Example

$$\varphi_1 = \{^0/_x, ^1/_y\} \text{ and } \varphi_2 = \{^1/_x, ^0/_y\}$$

$$\varphi_1 \not\sim_s \varphi_2 \text{ as } (x = 0)\varphi_1 \text{ while } (x \neq 0)\varphi_2.$$

Example

$$\varphi_1 = \nu k\{^{\mathsf{aenc}(0,\mathsf{pk}(k))}/_x, ^{\mathsf{pk}(k)}/_y\} \quad \varphi_2 = \nu k\{^{\mathsf{aenc}(1,\mathsf{pk}(k))}/_x, ^{\mathsf{pk}(k)}/_y\}$$

$$\varphi_1 \not\sim_s \varphi_2 \text{ as } (\mathsf{aenc}(0, y) = x)\varphi_1 \text{ while } (aenc(0, y) \neq x)\varphi_2$$

Static equivalence: examples

Example

$$\varphi_1 = \{^0/_x, ^1/_y\} \text{ and } \varphi_2 = \{^1/_x, ^0/_y\}$$

$$\varphi_1 \not\sim_s \varphi_2 \text{ as } (x = 0)\varphi_1 \text{ while } (x \neq 0)\varphi_2.$$

Example

$$\varphi_1 = \nu k\{^{\mathsf{aenc}(0,\mathsf{pk}(k))}/_x, ^{\mathsf{pk}(k)}/_y\} \qquad \varphi_2 = \nu k\{^{\mathsf{aenc}(1,\mathsf{pk}(k))}/_x, ^{\mathsf{pk}(k)}/_y\}$$

$$\varphi_1 \not\sim_s \varphi_2 \text{ as } (\mathsf{aenc}(0, y) = x)\varphi_1 \text{ while } (aenc(0, y) \neq x)\varphi_2$$

Need to model **randomisation** of encryption.

$$\varphi_1 = \nu k, r\{^{\mathsf{aenc}(0,r,\mathsf{pk}(k))}/_x, ^{\mathsf{pk}(k)}/_y\}$$
$$\varphi_2 = \nu k, r\{^{\mathsf{aenc}(1,r,\mathsf{pk}(k))}/_x, ^{\mathsf{pk}(k)}/_y\}$$

Then $\varphi_1' \sim_s \varphi_2'$.

Semantics of the applied pi calculus

Before defining indistinguishability of processes, we need a precise semantics!

A configuration is a triple:

$$(\mathcal{E}, \mathcal{P}, \varphi)$$

- $\mathcal{E}$ is the set of restricted names;
- $\mathcal{P}$ is the multiset of processes executed in parallel;
- φ is the frame of output messages
  (ignored in internal reduction)

Semantics of the applied pi calculus

Before defining indistinguishability of processes, we need a precise semantics!

A configuration is a triple:

$$(\mathcal{E}, \mathcal{P}, \varphi)$$

- $\mathcal{E}$ is the set of restricted names;
- $\mathcal{P}$ is the multiset of processes executed in parallel;
- φ is the frame of output messages (ignored in internal reduction)

Initial configuration for process P: $(\emptyset, \{\!\{P\}\!\}, \emptyset)$

Operational semantics: internal reduction

Internal reduction $\rightarrow$ is defined by rules (selection):

$$(\mathcal{E}, \mathcal{P} \cup \{\!\{0\}\!\}) \xrightarrow{\varepsilon} (\mathcal{E}, \mathcal{P}) \qquad (\textsc{Null})$$

$$(\mathcal{E}, \mathcal{P} \cup \{\!\{P \mid Q\}\!\}) \xrightarrow{\varepsilon} (\mathcal{E}, \mathcal{P} \cup \{\!\{P, Q\}\!\}) \qquad (\textsc{Par})$$

$$(\mathcal{E}, \mathcal{P} \cup \{\!\{\mathsf{new}\ n.P\}\!\}) \xrightarrow{\varepsilon} (\mathcal{E} \cup \{n'\}, \mathcal{P}\{^{n'}/_{n}\}) \qquad (\textsc{New})$$

$$\text{if } n' \text{ fresh}$$

$$(\mathcal{E}, \mathcal{P} \cup \{\!\{\mathsf{if}\ u = v\ \mathsf{then}\ P\ \mathsf{else}\ Q\}\!\}) \xrightarrow{\varepsilon} (\mathcal{E}, \mathcal{P} \cup \{\!\{P\}\!\}) \qquad (\textsc{Then})$$

$$\text{if } u =_E v$$

$$(\mathcal{E}, \mathcal{P} \cup \{\!\{\overline{u}\langle t\rangle.P, v(x).Q\}\!\}) \xrightarrow{\varepsilon} (\mathcal{E}, \mathcal{P} \cup \{\!\{P, Q\{^{t}/_{x}\}\}\!\}) \qquad (\textsc{Comm})$$

$$u =_E v$$

Indistinguishability as a process equivalence

Naturally modelled using **equivalences** from process calculi

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

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

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

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.

Example

$$\begin{aligned} P = \quad & \text{new } k.c(x).\overline{c}\langle \text{enc}(x,k)\rangle.\overline{c}\langle k\rangle \\ Q = \quad & \text{new } k.c(x).\overline{c}\langle \text{enc}(0,k)\rangle.\overline{c}\langle k\rangle \end{aligned}$$

$P \not\approx Q$ as $A|P \Downarrow d$, but $A|Q \not\Downarrow d$ for

$$A = \overline{c}\langle 1\rangle.c(y).c(z).\,\text{if}\,\text{dec}(y,z) = 1\,\text{then}\,\overline{d}\langle 1\rangle$$

Labelled semantics

Reasoning about **all** processes A not convenient.

Extend $\xrightarrow{\epsilon}$ to directly interact with a (non specified) adversary.

$$(\mathcal{E}, \mathcal{P} \cup \{\!\{u(x).P\}\!\}, \Phi) \xrightarrow{\xi(\zeta)} (\mathcal{E}, \mathcal{P} \cup \{\!\{P\{^{\zeta\Phi}/_{x}\}\}\!\}, \Phi) \qquad \text{(In)}$$
$$\text{if } \nu\mathcal{E}.\Phi \vdash^{\xi} u$$

$$(\mathcal{E}, \mathcal{P} \cup \{\!\{\overline{u}\langle t\rangle.P\}\!\}, \Phi) \xrightarrow{\overline{\xi}\langle \mathsf{ax}_n\rangle} (\mathcal{E}, \mathcal{P} \cup \{\!\{P\}\!\}, \Phi \cup \{^{t}/_{\mathsf{ax}_n}\}) \qquad \text{(Out)}$$
$$\text{if } \nu\mathcal{E}.\Phi \vdash^{\xi} u \text{ and } n = |\Phi| + 1$$

Labelled semantics

Reasoning about **all** processes A not convenient.

Extend $\xrightarrow{\epsilon}$ to directly interact with a (non specified) adversary.

$$(\mathcal{E}, \mathcal{P} \cup \{\!\{u(x).P\}\!\}, \Phi) \xrightarrow{\xi(\zeta)} (\mathcal{E}, \mathcal{P} \cup \{\!\{P\{^{\zeta\Phi}/_x\}\}\!\}, \Phi) \quad \text{if } \nu\mathcal{E}.\Phi \vdash^{\xi} u \qquad (\textsc{In})$$

$$(\mathcal{E}, \mathcal{P} \cup \{\!\{\overline{u}\langle t\rangle.P\}\!\}, \Phi) \xrightarrow{\overline{\xi}\langle \mathsf{ax}_n\rangle} (\mathcal{E}, \mathcal{P} \cup \{\!\{P\}\!\}, \Phi \cup \{^{t}/_{\mathsf{ax}_n}\}) \quad \text{if } \nu\mathcal{E}.\Phi \vdash^{\xi} u \text{ and } n = |\Phi| + 1 \qquad (\textsc{Out})$$

Example

$$P = \quad \mathsf{new}\ k.c(x).\overline{c}\langle \mathsf{enc}(x, k)\rangle.\overline{c}\langle k\rangle$$

$$(\emptyset, P, \emptyset) \xRightarrow{c(1)} \xRightarrow{\overline{c}\langle \mathsf{ax}_1\rangle} \xRightarrow{\overline{c}\langle \mathsf{ax}_2\rangle} (\{k'\}, \emptyset, \{^{\mathsf{enc}(1,k')}/_{\mathsf{ax}_1}, ^{k'}/_{\mathsf{ax}_2}\})$$

where $\xRightarrow{\ell} = \xrightarrow{\epsilon}{}^{*} \xrightarrow{\ell} \xrightarrow{\epsilon}{}^{*}$.

Indistinguishability using labelled semantics

Trace equivalence

$$P \approx_t Q$$

iff

$$\text{if } P \stackrel{\mathsf{tr}}{\Rightarrow} (\mathcal{E}, \mathcal{P}, \varphi) \text{ then } Q \stackrel{\mathsf{tr}}{\Rightarrow} (\mathcal{E}', \mathcal{Q}, \varphi') \wedge \varphi \sim_s \varphi' \text{ for some } \mathcal{E}', \mathcal{Q}, \varphi'$$

(and vice-versa)

Intuition:
Same adversary behaviour (tr) yields indistinguishable frames ($\sim_s$)

Indistinguishability using labelled semantics

Trace equivalence

$$P \approx_t Q$$

iff

$$\text{if } P \overset{\text{tr}}{\Rightarrow} (\mathcal{E}, \mathcal{P}, \varphi) \text{ then } Q \overset{\text{tr}}{\Rightarrow} (\mathcal{E}', \mathcal{Q}, \varphi') \wedge \varphi \sim_s \varphi' \text{ for some } \mathcal{E}', \mathcal{Q}, \varphi'$$

(and vice-versa)

Intuition:

Same adversary behaviour (tr) yields indistinguishable frames ($\sim_s$)

Example

$$P = \text{new } k.c(x).\overline{c}\langle \text{enc}(x,k)\rangle.\overline{c}\langle k\rangle$$
$$Q = \text{new } k.c(x).\overline{c}\langle \text{enc}(0,k)\rangle.\overline{c}\langle k\rangle$$

$$P \not\approx_t Q \text{ as } \begin{array}{ll} P \xRightarrow{c(1)\,\overline{c}\langle \text{ax}_1\rangle\,\overline{c}\langle \text{ax}_2\rangle} & (\{k'\}, \emptyset, \{^{\text{enc}(1,k')}/_{\text{ax}_1}, ^{k'}/_{\text{ax}_2}\}) \\ & \qquad\qquad \not\sim_s \\ Q \xRightarrow{c(1)\,\overline{c}\langle \text{ax}_1\rangle\,\overline{c}\langle \text{ax}_2\rangle} & (\{k'\}, \emptyset, \{^{\text{enc}(0,k')}/_{\text{ax}_1}, ^{k'}/_{\text{ax}_2}\}) \end{array}$$

Indistinguishability using labelled semantics

Trace equivalence

$$P \approx_t Q$$

iff

$$\text{if } P \overset{\mathsf{tr}}{\Rightarrow} (\mathcal{E}, \mathcal{P}, \varphi) \text{ then } Q \overset{\mathsf{tr}}{\Rightarrow} (\mathcal{E}', \mathcal{Q}, \varphi') \land \varphi \sim_s \varphi' \text{ for some } \mathcal{E}', \mathcal{Q}, \varphi'$$

(and vice-versa)

Intuition:

Same adversary behaviour (tr) yields indistinguishable frames ($\sim_s$)

Theorem

$$P \approx_t Q \implies P \approx Q$$

A tour to the (equivalence) zoo

$\boxed{\text{testing equiv.}} \Longleftarrow \boxed{\text{obs. equiv.}}$

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

$$\begin{tikzcd}
\boxed{\text{testing equiv.}} & \boxed{\text{obs. equiv.}} \arrow[l, Rightarrow] \arrow[d, Leftrightarrow] \\
 & \boxed{\text{labelled bisim.}}
\end{tikzcd}$$

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

A tour to the (equivalence) zoo

$$\begin{tikzcd}
 & \boxed{\text{diff equiv.}} \arrow[d, Rightarrow] \\
\boxed{\text{testing equiv.}} & \boxed{\text{obs. equiv.}} \arrow[l, Rightarrow] \arrow[d, Leftrightarrow] \\
 & \boxed{\text{labelled bisim.}}
\end{tikzcd}$$

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

A tour to the (equivalence) zoo

$$\begin{tikzcd}
& \text{diff equiv.} \arrow[d, Rightarrow] \\
\text{testing equiv.} & \text{obs. equiv.} \arrow[l, Rightarrow] \arrow[d, Leftrightarrow] \\
& \text{labelled bisim.}
\end{tikzcd}$$

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

$$\begin{tikzcd}
 & \text{diff equiv.} \arrow[d, Rightarrow] & \\
\text{testing equiv.} & \text{obs. equiv.} \arrow[l, Rightarrow] \arrow[d, Leftrightarrow] & \\
 & \text{labelled bisim.} & \text{symbolic bisim.} \arrow[l, Rightarrow]
\end{tikzcd}$$

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

A tour to the (equivalence) zoo

$$
\begin{tikzcd}
& \text{diff equiv.} \arrow[d, Rightarrow] & \\
\text{testing equiv.} & \text{obs. equiv.} \arrow[l, Rightarrow] \arrow[d, Leftrightarrow] & \\
& \text{labelled bisim.} \arrow[r, Leftrightarrow] & \text{symbolic bisim.}
\end{tikzcd}
$$

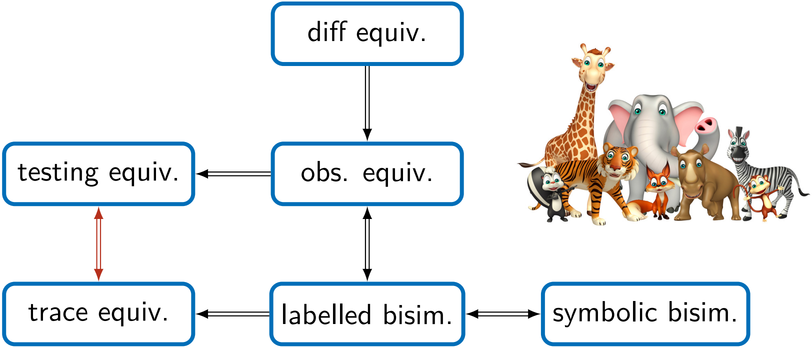
Liu, Lin. A complete symbolic bisimulation for full applied pi calculus.TCS'12

A tour to the (equivalence) zoo

$$\begin{tikzcd}
 & \text{diff equiv.} \arrow[d, Rightarrow] & \\
\text{testing equiv.} & \text{obs. equiv.} \arrow[l, Rightarrow] \arrow[d, Leftrightarrow] & \\
\text{trace equiv.} \arrow[u, Rightarrow] & \text{labelled bisim.} \arrow[l, Rightarrow] \arrow[r, leftrightarrow] & \text{symbolic bisim.}
\end{tikzcd}$$

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).

A tour to the (equivalence) zoo

$$\begin{tikzcd}
 & \text{diff equiv.} \arrow[d, Rightarrow] & \\
\text{testing equiv.} \arrow[r, Leftrightarrow] \arrow[d, Leftrightarrow] & \text{obs. equiv.} \arrow[d, Leftrightarrow] & \\
\text{trace equiv.} \arrow[r, Leftrightarrow] & \text{labelled bisim.} \arrow[r, Leftrightarrow] & \text{symbolic bisim.}
\end{tikzcd}$$

For a class of **determinate processes**.

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

Part II

Applications: modelling security protocols

Secrecy in symbolic models

Symbolic analysis: secrecy generally modelled as **non-deducibility**:

the attacker cannot compute the value of the secret

$\rightsquigarrow$ **partial leakage** not detected

Example

Let h be a one-way hash function. The protocol

$$P = \text{new } s.out(c, h(s))$$

would be considered to enforce the secrecy of s.

Secrecy as indistinguishability

Stronger notions of secrecy can be defined using indistinguishability

- **Strong secrecy** of s: [Blanchet'04]

$$\mathbf{in}(c, \langle t_1, t_2 \rangle).\ P\{^{t_1}/_s\} \approx \mathbf{in}(c, \langle t_1, t_2 \rangle).\ P\{^{t_2}/_s\}$$

Even if the attacker chooses values t_1 or t_2 he cannot distinguish whether t_1 or t_2 was used as the secret.

- **Real-or-random**

$$P; \mathbf{out}(s) \approx P; \text{new } s'.\mathbf{out}(s')$$

The attacker cannot distinguish whether at the end of the protocol he is given the **real** *secret or a* **random** *value.*

$\rightsquigarrow$ Resistance against **offline guessing attacks**

Modelling resistance against offline guessing attacks

$\nu w.\phi$ is **resistant to guessing attacks** against w iff

$$\nu w.(\phi \cup \{^{w}/_{x}\}) \sim_s \nu w, w'.(\phi \cup \{^{w'}/_{x}\})$$

Intuition: an attacker cannot distinguish the right guess from a wrong guess

A process P is resistant against guessing attacks on w if whenever

$$(\{w\}, \{\!\{P\}\!\}, \emptyset) \overset{\ell}{\Rightarrow}^{*} (\mathcal{E}, \mathcal{P}, \varphi)$$

then φ is resistant to guessing attacks.

Example: EKE protocol [BellovinMerritt92]

$$
\begin{array}{lll}
\mathrm{A} \to \mathrm{B}: & \mathsf{enc}(\mathsf{pk}(k), w) & (\text{EKE.1}) \\
\mathrm{B} \to \mathrm{A}: & \mathsf{enc}(\mathsf{aenc}(r, \mathsf{pk}(k)), w) & (\text{EKE.2}) \\
\mathrm{A} \to \mathrm{B}: & \mathsf{enc}(na, r) & (\text{EKE.3}) \\
\mathrm{B} \to \mathrm{A}: & \mathsf{enc}(\langle na, nb \rangle, r) & (\text{EKE.4}) \\
\mathrm{A} \to \mathrm{B}: & \mathsf{enc}(nb, r) & (\text{EKE.5})
\end{array}
$$

$$
\phi = \nu k, r, na, nb. \quad \{^{\mathsf{enc}(\mathsf{pk}(k),w)}/_{x_1}, ^{\mathsf{enc}(\mathsf{aenc}(r,\mathsf{pk}(k)),w)}/_{x_2}, ^{\mathsf{enc}(na,r)}/_{x_3}, ^{\mathsf{enc}(\langle na,nb\rangle,r)}/_{x_4}, ^{\mathsf{enc}(nb,r)}/_{x_5}\}
$$

$$
\nu w.(\phi \cup \{^{w}/_{x}\}) \overset{?}{\sim}_s \nu w, w'.(\phi \cup \{^{w'}/_{x})\}
$$

Example: EKE protocol [BellovinMerritt92]

$$
\begin{array}{lll}
\mathrm{A} \rightarrow \mathrm{B}: & \mathsf{enc}(\mathsf{pk}(k), w) & (\mathsf{EKE.1}) \\
\mathrm{B} \rightarrow \mathrm{A}: & \mathsf{enc}(\mathsf{aenc}(r, \mathsf{pk}(k)), w) & (\mathsf{EKE.2}) \\
\mathrm{A} \rightarrow \mathrm{B}: & \mathsf{enc}(na, r) & (\mathsf{EKE.3}) \\
\mathrm{B} \rightarrow \mathrm{A}: & \mathsf{enc}(\langle na, nb \rangle, r) & (\mathsf{EKE.4}) \\
\mathrm{A} \rightarrow \mathrm{B}: & \mathsf{enc}(nb, r) & (\mathsf{EKE.5})
\end{array}
$$

$$
\phi = \nu k, r, na, nb. \quad \{ ^{\mathsf{enc}(\mathsf{pk}(k),w)}/_{x_1}, ^{\mathsf{enc}(\mathsf{aenc}(r,\mathsf{pk}(k)),w)}/_{x_2}, ^{\mathsf{enc}(na,r)}/_{x_3}, ^{\mathsf{enc}(\langle na,nb\rangle,r)}/_{x_4}, ^{\mathsf{enc}(nb,r)}/_{x_5} \}
$$

$$
\nu w.(\phi \cup \{^{w}/_{x}\}) \overset{?}{\sim}_s \nu w, w'.(\phi \cup \{^{w'}/_{x})\}
$$

- holds if we suppose the equation $\mathsf{enc}(\mathsf{dec}(x, y), y) = x$ otherwise the test $\mathsf{enc}(\mathsf{dec}(x_1, x)) \overset{?}{=}_{\mathcal{E}} x_1$ distinguishes

Example: EKE protocol [BellovinMerritt92]

$$\begin{array}{llr}
\mathrm{A} \rightarrow \mathrm{B}: & \mathsf{enc}(\mathsf{pk}(k), w) & (\mathsf{EKE.1}) \\
\mathrm{B} \rightarrow \mathrm{A}: & \mathsf{enc}(\mathsf{aenc}(r, \mathsf{pk}(k)), w) & (\mathsf{EKE.2}) \\
\mathrm{A} \rightarrow \mathrm{B}: & \mathsf{enc}(na, r) & (\mathsf{EKE.3}) \\
\mathrm{B} \rightarrow \mathrm{A}: & \mathsf{enc}(\langle na, nb \rangle, r) & (\mathsf{EKE.4}) \\
\mathrm{A} \rightarrow \mathrm{B}: & \mathsf{enc}(nb, r) & (\mathsf{EKE.5})
\end{array}$$

$$\phi = \nu k, r, na, nb. \quad \{^{\mathsf{enc}(\mathsf{pk}(k),w)}/_{x_1}, {}^{\mathsf{enc}(\mathsf{aenc}(r,\mathsf{pk}(k)),w)}/_{x_2}, {}^{\mathsf{enc}(na,r)}/_{x_3}, {}^{\mathsf{enc}(\langle na,nb\rangle,r)}/_{x_4}, {}^{\mathsf{enc}(nb,r)}/_{x_5}\}$$

$$\nu w.(\phi \cup \{^{w}/_{x}\}) \ \overset{?}{\sim}_s \ \nu w, w'.(\phi \cup \{^{w'}/_{x})\}$$

- holds if we suppose the equation $\mathsf{enc}(\mathsf{dec}(x, y), y) = x$ otherwise the test $\mathsf{enc}(\mathsf{dec}(x_1, x)) \overset{?}{=}_{\mathcal{E}} x_1$ distinguishes
- if we add equation $\mathsf{ispubkey}(\mathsf{pk}(x)) = \mathsf{ok}$ we distinguish frames by $\mathsf{ispubkey}(\mathsf{dec}(x_1, x)) \overset{?}{=}_{\mathcal{E}} \mathsf{ok}$

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:~~
  $\cancel{V_A(v) \approx V_B(v)}$

- 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:~~
  $\cancel{V_A(v) \approx V_B(v)}$
- 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~~: $\cancel{V_A(v) \approx V_B(v)}$
- ~~The attacker cannot distinguish A votes 0 and A votes 1~~: $\cancel{V_A(0) \approx V_A(1)}$
- 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

The Helios e-voting protocol (MixNet version)

authenticated channel

BB

$$V_1 \xrightarrow{\langle id_1,\, aenc(\mathsf{pk}_E,\, r_1,\, v_1)\rangle} \quad \boxed{\langle id_1, \mathsf{aenc}(\mathsf{pk}_E, r_1, v_1)\rangle}$$

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

The Helios e-voting protocol (MixNet version)

authenticated channel

$$V_1 \xrightarrow{\langle id_1, aenc(\mathrm{pk}_E, r_1, v_1)\rangle}$$

$$V_2 \xrightarrow{\langle id_2, aenc(\mathrm{pk}_E, r_2, v_2)\rangle}$$

BB

$$\langle id_1, \mathrm{aenc}(\mathrm{pk}_E, r_1, v_1)\rangle$$

$$\langle id_2, aenc(\mathrm{pk}_E, r_2, v_2)\rangle$$

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

The Helios e-voting protocol (MixNet version)

authenticated channel

BB

$$V_1 \xrightarrow{\langle id_1,\, aenc(\mathrm{pk}_E,\, r_1,\, v_1)\rangle} \quad \langle id_1, \mathrm{aenc}(\mathrm{pk}_E, r_1, v_1)\rangle$$

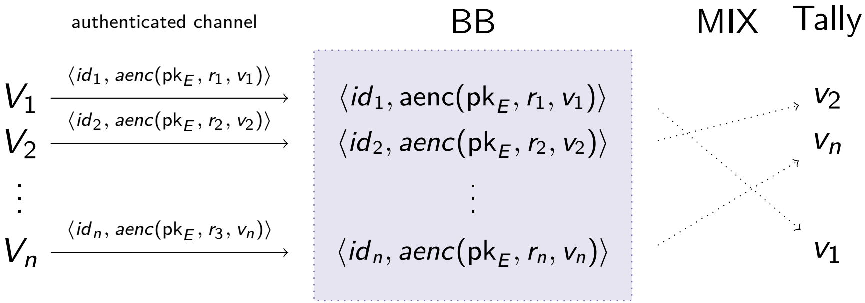
$$V_2 \xrightarrow{\langle id_2,\, aenc(\mathrm{pk}_E,\, r_2,\, v_2)\rangle} \quad \langle id_2, aenc(\mathrm{pk}_E, r_2, v_2)\rangle$$

$$\vdots \qquad\qquad\qquad\qquad \vdots$$

$$V_n \xrightarrow{\langle id_n,\, aenc(\mathrm{pk}_E,\, r_3,\, v_n)\rangle} \quad \langle id_n, aenc(\mathrm{pk}_E, r_n, v_n)\rangle$$

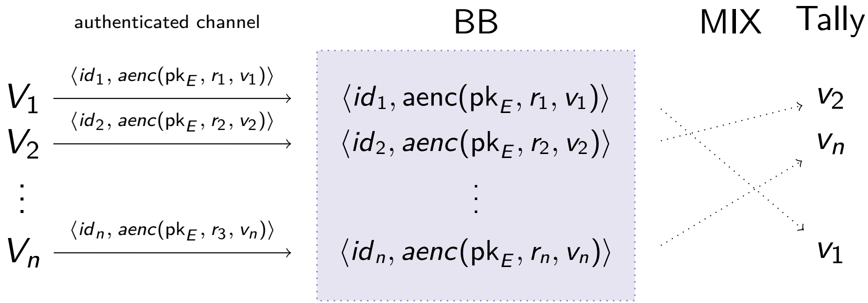
where pk_E is the election public key and MIX a verifiable mixnet.

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

The Helios e-voting protocol (MixNet version)

authenticated channel | BB | MIX | Tally

$$V_1 \xrightarrow{\langle id_1,\, aenc(\mathrm{pk}_E,\, r_1,\, v_1)\rangle} \quad \langle id_1, \mathrm{aenc}(\mathrm{pk}_E, r_1, v_1)\rangle \qquad v_2$$

$$V_2 \xrightarrow{\langle id_2,\, aenc(\mathrm{pk}_E,\, r_2,\, v_2)\rangle} \quad \langle id_2, aenc(\mathrm{pk}_E, r_2, v_2)\rangle \qquad v_n$$

$$\vdots \qquad\qquad\qquad \vdots$$

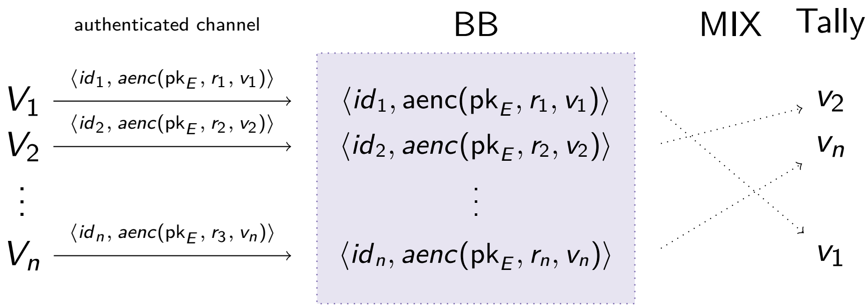
$$V_n \xrightarrow{\langle id_n,\, aenc(\mathrm{pk}_E,\, r_3,\, v_n)\rangle} \quad \langle id_n, aenc(\mathrm{pk}_E, r_n, v_n)\rangle \qquad v_1$$

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!**

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: Helios$(v_1, v_2) \overset{?}{\approx}_t$ 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: $\mathsf{break}(\mathsf{aenc}(\mathsf{pk}(x), y, z)) \to z$
- 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, ...

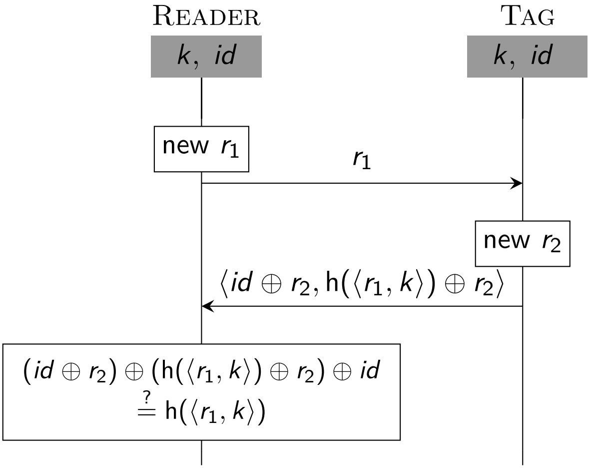
Can be modelled as an equivalence property:

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

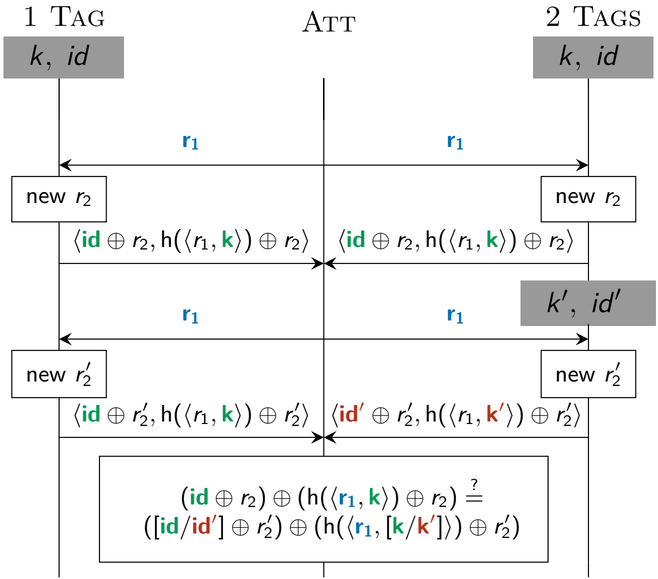
Authentication protocol of a RFID tag (KCL)



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



Part III

Automated analysis : ProVerif

Bi-processes

We want to prove $P_1 \approx P_2$.

In ProVerif P_1 and P_2 are jointly specified using a **bi-process**.
using the **choice[t_1, t_2]** operator.

When P contains **choice**$[t_1, t_2]$:

- P_1 is defined by replacing choice$[t_1, t_2]$ by t_1;
- P_2 is defined by replacing choice$[t_1, t_2]$ by t_2.

Bi-processes

We want to prove $P_1 \approx P_2$.

In ProVerif P_1 and P_2 are jointly specified using a **bi-process**. using the **choice[**t_1, t_2**]** operator.

When P contains **choice**$[t_1, t_2]$:

- P_1 is defined by replacing choice$[t_1, t_2]$ by t_1;
- P_2 is defined by replacing choice$[t_1, t_2]$ by t_2.

Remark: Any process can be defined as a bi-process:

$$\text{if choice}[0,1] = 0 \text{ then } P_1 \text{ else } P_2$$

but ProVerif does not succeed on general processes

Diff-equivalence

Diff equivalence is a fine-grained equivalence that implies trace equivalence

$\mathbf{P} \approx_{\text{diff}} \mathbf{Q}$: taking the same branches in P and Q implies static equivalence ($\sim$ reachability $+$ static equivalence).

Diff-equivalence

Diff equivalence is a fine-grained equivalence that implies trace equivalence

$P \approx_{\text{diff}} Q$: taking the same branches in P and Q implies static equivalence ($\sim$ reachability $+$ static equivalence).

Often **too fine-grained**:

$$\overline{c}\langle \text{choice}[a, b]\rangle | \overline{c}\langle \text{choice}[b, a]\rangle$$ **not** considered equivalent!

Diff-equivalence

Diff equivalence is a fine-grained equivalence that implies trace equivalence

$\mathbf{P} \approx_{\mathrm{diff}} \mathbf{Q}$: taking the same branches in P and Q implies static equivalence ($\sim$ reachability $+$ static equivalence).

Often **too fine-grained**:

$$\overline{c}\langle \mathrm{choice}[a, b]\rangle | \overline{c}\langle \mathrm{choice}[b, a]\rangle \text{ **not** considered equivalent!}$$

Recent versions include a command **equivalence**, constructing the bi-process automatically:

$$P = \overline{c}\langle a\rangle.\overline{c}\langle b\rangle \parallel \overline{c}\langle b\rangle.\overline{c}\langle a\rangle$$
$$Q = \overline{c}\langle b\rangle.\overline{c}\langle a\rangle \parallel \overline{c}\langle a\rangle.\overline{c}\langle b\rangle$$
$$\text{equivalence } P\ Q$$

fails

Strong flavors of secrecy

Strong secrecy (non-interference) of x

$$c(x_1).c(x_2).P\{x \mapsto \mathrm{choice}[x_1, x_2]\}$$

(or direct query `noninterf x`)

Resistance to guessing attacks of w

$$\mathrm{new}\ w.P.\mathrm{new}\ w'.\overline{c}\langle choice[w, w']\rangle\}$$

(or direct query `weaksecret w`)

Modelling equivalence in Horn clauses

Reachability properties: $\mathrm{att}(t)$ models attacker knows t

Equivalence properties:
$\mathrm{att}'(t_1, t_2)$ models attacker knows t_1 in P_1 and t_2 in P_2

$\overline{c}\langle \mathrm{choice}[t_1, t_2] \rangle$ is translated into $\mathrm{att}'(t_1, t_2)$

Special clauses for equivalence

$$\mathrm{att}'(x, y) \wedge \mathrm{att}'(x, y') \wedge \mathrm{nounif}(y, y') \rightarrow \mathrm{bad}$$
$$\mathrm{att}'(x, y) \wedge \mathrm{att}'(x', y) \wedge \mathrm{nounif}(x, x') \rightarrow \mathrm{bad}$$

where $\mathrm{nounif}(t, t')$ holds when t, t' cannot be unified.

Equivalence holds when bad cannot be derived.

Part IV

Automated analysis : DEEPSEC

DEEPSEC: DEciding Equivalence Properties in SECurity protocols

- **Decision procedure** for trace equivalence
  (no approximation, but high complexity coNEXP!)
- **Bounded number of sessions**
  (no replication; otherwise full applied pi)
- Crypto primitives specified by
  destructor subterm convergent rewrite systems

- Tool implemented in OCaml:
  `https://github.com/DeepSec-prover/deepsec`
- Input language similar to (untyped) ProVerif
- Possibility to distribute the verification
  (on multiple cores and multiple machines)

Destructor subterm convergent rewrite systems

Rewrite rules orient equational theories : $\ell \to r$ rather than $\ell = r$.

- Partition function symbols into constructors and destructors
- Messages do not contain destructors
- Each destructor g defined by rules $g(t_1, \ldots, t_n) \to u$
- For any rule $\ell \to r$ r is a subterm of ℓ (or constant)

Example

$$\mathcal{F}_c = \{\mathrm{enc}, \mathrm{pair}\} \quad \mathcal{F}_d = \{\mathrm{dec}, \mathrm{fst}, \mathrm{snd}\}$$
$$\mathcal{R} = \{\mathrm{dec}(enc(x, y), y) \to x, \mathrm{fst}(\mathrm{pair}(x, y)) \to x, \mathrm{snd}(\mathrm{pair}(x, y)) \to y\}$$

$\mathrm{dec}(\mathrm{pair}(t_1, t_2))$ not a valid message!

Verification for a bounded number of sessions

Bounded number of sessions: why is it difficult?

Verification for a bounded number of sessions

Bounded number of sessions: why is it difficult?

The state space is still **infinite**: unbounded number of attacker inputs!

Verification for a bounded number of sessions

Bounded number of sessions: why is it difficult?

The state space is still **infinite**: unbounded number of attacker inputs!

Idea: represent infinite number of possible inputs **symbolically** in a **constraint system**

Verification for a bounded number of sessions

Bounded number of sessions: why is it difficult?

The state space is still **infinite**: unbounded number of attacker inputs!

Idea: represent infinite number of possible inputs **symbolically** in a **constraint system**

Example

$c(x).P$ transitions to P but keeps a deduction constraint $X \vdash^? x$

Verification for a bounded number of sessions

Bounded number of sessions: why is it difficult?

The state space is still **infinite**: unbounded number of attacker inputs!

Idea: represent infinite number of possible inputs **symbolically** in a **constraint system**

Example

$c(x).P$ transitions to P but keeps a deduction constraint $X \vdash^? x$

if $t_1 = t_2$ then P else Q : 2 transitions

- to P with constraint $t_1 =^?_{\mathcal{R}} t_2$
- to Q with constraint $t_1 \neq^?_{\mathcal{R}} t_2$

Constraint systems

A **constraint system** is a tuple $\mathcal{C} = (\Phi, \mathsf{D}, \mathsf{E}^1)$ where:

- $\Phi = \{\mathsf{ax}_1 \mapsto t_1, \ldots, \mathsf{ax}_n \mapsto t_n\}$ is a frame;
- D is a conjunction of deduction facts $X \vdash^? x$;
- E^1 is a conjunction of formulas $u =^?_{\mathcal{R}} v$ or $u \neq^?_{\mathcal{R}} v$.

A **solution** is a pair of substitutions Σ, σ such that

- $\Phi\sigma \vdash^{X\Sigma} x\sigma$ for all $X \vdash^? x \in \mathsf{D}$
- $u\sigma \bowtie v\sigma$ for all $u \bowtie v \in \mathsf{E}^1$

Note: Σ represents attacker inputs and constraints are such that it completely defines σ

Symbolic semantics

Symbolic semantics: associate a constraint system to the process (sample rules)

$$(\mathcal{P} \cup \{\!\{\text{if } u = v \text{ then } Q\}\!\}, (\Phi, \mathsf{D}, \mathsf{E}^1)) \xrightarrow{\varepsilon}_{\mathsf{s}} (\mathcal{P} \cup \{\!\{Q\}\!\}, (\Phi, \mathsf{D}, \mathsf{E}^1 \wedge u =^?_{\mathcal{R}} v))$$

$$(\mathcal{P} \cup \{\!\{c(x).Q\}\!\}, (\Phi, \mathsf{D}, \mathsf{E}^1)) \xrightarrow{c(X)}_{\mathsf{s}} (\mathcal{P} \cup \{\!\{Q\}\!\}, (\Phi, \mathsf{D} \wedge X \vdash^? x, \mathsf{E}^1))$$

$$(\mathcal{P} \cup \{\!\{\overline{c}\langle t\rangle.Q\}\!\}, (\Phi, \mathsf{D}, \mathsf{E}^1)) \xrightarrow{\overline{c}\langle \mathsf{ax}\rangle}_{\mathsf{s}} (\mathcal{P} \cup \{\!\{Q\}\!\}, (\Phi \cup \{\mathsf{ax} \mapsto t\}, \mathsf{D}, \mathsf{E}^1))$$

Symbolic semantics

Symbolic semantics: associate a constraint system to the process (sample rules)

$$(\mathcal{P} \cup \{\!\{\text{if } u = v \text{ then } Q\}\!\}, (\Phi, \mathsf{D}, \mathsf{E}^1)) \xrightarrow{\varepsilon}_{\mathsf{s}} (\mathcal{P} \cup \{\!\{Q\}\!\}, (\Phi, \mathsf{D}, \mathsf{E}^1 \wedge u =^?_{\mathcal{R}} v))$$

$$(\mathcal{P} \cup \{\!\{c(x).Q\}\!\}, (\Phi, \mathsf{D}, \mathsf{E}^1)) \xrightarrow{c(X)}_{\mathsf{s}} (\mathcal{P} \cup \{\!\{Q\}\!\}, (\Phi, \mathsf{D} \wedge X \vdash^? x, \mathsf{E}^1))$$

$$(\mathcal{P} \cup \{\!\{\overline{c}\langle t\rangle.Q\}\!\}, (\Phi, \mathsf{D}, \mathsf{E}^1)) \xrightarrow{\overline{c}\langle \mathsf{ax}\rangle}_{\mathsf{s}} (\mathcal{P} \cup \{\!\{Q\}\!\}, (\Phi \cup \{\mathsf{ax} \mapsto t\}, \mathsf{D}, \mathsf{E}^1))$$

Sound: if $(A, \mathcal{C}) \xrightarrow{\ell}_{\mathsf{s}} (A', \mathcal{C}')$ then for any $(\Sigma, \sigma) \in Sol(\mathcal{C})$ we have that $A\sigma \xrightarrow{\ell\Sigma} A'\sigma$

Complete: if $(\Sigma, \sigma) \in Sol(\mathcal{C})$ and $A\sigma \xrightarrow{\ell\Sigma} A'$ then $(A, \mathcal{C}) \xrightarrow{\ell}_{\mathsf{s}} (A', \mathcal{C}')$ and $\Sigma', \sigma' \in Sol(C')$ and $A''\sigma' = A'$

A simple example

$$P^b \triangleq c(x).\,\mathsf{if}\ x = b\ \mathsf{then}\ \overline{c}\langle 0\rangle\ \mathsf{else}\ \overline{c}\langle x\rangle \qquad\qquad b \in \{0,1\}$$

$$Q \triangleq c(x).\overline{c}\langle x\rangle$$

A simple example

$$
\begin{aligned}
P^b &\triangleq c(x).\,\mathsf{if}\ x = b\ \mathsf{then}\ \overline{c}\langle 0\rangle\ \mathsf{else}\ \overline{c}\langle x\rangle \qquad\qquad b \in \{0,1\}\\
Q &\triangleq c(x).\overline{c}\langle x\rangle
\end{aligned}
$$

$P^0 \approx_t Q$ but $P^1 \not\approx_t Q$ (different behavior on input 1)

A simple example

$$P^b \triangleq c(x).\,\mathsf{if}\ x = b\ \mathsf{then}\ \overline{c}\langle 0\rangle\ \mathsf{else}\ \overline{c}\langle x\rangle \qquad\qquad b \in \{0,1\}$$
$$Q \triangleq c(x).\overline{c}\langle x\rangle$$

$P^0 \approx_t Q$ but $P^1 \not\approx_t Q$ (different behavior on input 1)

Symbolic transitions tree:

$$(\mathcal{P}_0^b, \mathcal{C}_\emptyset) \xrightarrow{c(X)}_{\mathsf{s}} (\mathcal{P}_1^b, \mathcal{C}_1^b) \begin{array}{l} \xrightarrow{\varepsilon}_{\mathsf{s}} (\mathcal{P}_2^b, \mathcal{C}_2^b) \xrightarrow{\overline{c}\langle \mathsf{ax}_1\rangle}_{\mathsf{s}} (\mathcal{P}_4^b, \mathcal{C}_4^b) \\ \xrightarrow{\varepsilon}_{\mathsf{s}} (\mathcal{P}_3^b, \mathcal{C}_3^b) \xrightarrow{\overline{c}\langle \mathsf{ax}_1\rangle}_{\mathsf{s}} (\mathcal{P}_5^b, \mathcal{C}_5^b) \end{array}$$

$$(\mathcal{Q}_0, \mathcal{C}_\emptyset) \xrightarrow{c(X)}_{\mathsf{s}} (\mathcal{Q}_1, \mathcal{C}_1) \xrightarrow{\qquad\quad \overline{c}\langle \mathsf{ax}_1\rangle \qquad\quad}_{\mathsf{s}} (\mathcal{Q}_2, \mathcal{C}_2)$$

$$\begin{array}{lcl} \mathcal{C}_2 & \triangleq & (\{\mathsf{ax}_1 \mapsto x\}, X \vdash^? x, \emptyset) \\ \mathcal{C}_4^b & \triangleq & (\{\mathsf{ax}_1 \mapsto 0\}, X \vdash^? x, x =_{\mathcal{R}}^? b) \\ \mathcal{C}_4^b & \triangleq & (\{\mathsf{ax}_1 \mapsto x\}, X \vdash^? x, x \neq_{\mathcal{R}}^? b) \end{array}$$

Partition Tree

Build a **joint** symbolic execution tree

Partition solutions (split nodes): ensure static equivalences of all solutions in a same node
$\rightsquigarrow$ done by **constraint solving algorithm**

$(\mathcal{Q}_0, \mathcal{C}_\emptyset)$
$(\mathcal{P}_0^0, \mathcal{C}_\emptyset)$

Partition Tree

Build a **joint** symbolic execution tree

Partition solutions (split nodes): ensure static equivalences of all solutions in a same node
$\rightsquigarrow$ done by **constraint solving algorithm**

$$\begin{matrix} (\mathcal{Q}_0, \mathcal{C}_\emptyset) \\ (\mathcal{P}_0^0, \mathcal{C}_\emptyset) \end{matrix} \xRightarrow{c(X)}_{\mathsf{s}} \begin{matrix} (\mathcal{Q}_1, \mathcal{C}_1), \ (\mathcal{P}_1^0, \mathcal{C}_1^0) \\ (\mathcal{P}_2^0, \mathcal{C}_2^0), \ (\mathcal{P}_3^0, \mathcal{C}_3^0) \end{matrix}$$

Partition Tree

Build a **joint** symbolic execution tree

Partition solutions (split nodes): ensure static equivalences of all solutions in a same node
$\rightsquigarrow$ done by **constraint solving algorithm**

$$\begin{matrix}(\mathcal{Q}_0, \mathcal{C}_\emptyset)\\(\mathcal{P}_0^0, \mathcal{C}_\emptyset)\end{matrix} \xRightarrow{c(X)}_{\mathsf{s}} \begin{matrix}(\mathcal{Q}_1, \mathcal{C}_1),\ (\mathcal{P}_1^0, \mathcal{C}_1^0)\\(\mathcal{P}_2^0, \mathcal{C}_2^0),\ (\mathcal{P}_3^0, \mathcal{C}_3^0)\end{matrix} \xRightarrow{\overline{c}\langle \mathsf{ax}_1 \rangle}_{\mathsf{s}} \begin{matrix}(\mathcal{Q}_2, \mathcal{C}_2),\\(\mathcal{P}_4^0, \mathcal{C}_4^0),\ (\mathcal{P}_5^0, \mathcal{C}_5^0)\end{matrix}$$

Need to **partition**: $\mathcal{C}_4^0$ enforces $X = 0$ and $\mathcal{C}_5^0$ enforces $X \neq 0$.

Partition Tree

Build a **joint** symbolic execution tree

Partition solutions (split nodes): ensure static equivalences of all solutions in a same node
$\rightsquigarrow$ done by **constraint solving algorithm**

$$\begin{array}{c}(\mathcal{Q}_0, \mathcal{C}_\emptyset)\\(\mathcal{P}^0_0, \mathcal{C}_\emptyset)\end{array} \xRightarrow{c(X)}_{s} \begin{array}{c}(\mathcal{Q}_1, \mathcal{C}_1),\ (\mathcal{P}^0_1, \mathcal{C}^0_1)\\(\mathcal{P}^0_2, \mathcal{C}^0_2),\ (\mathcal{P}^0_3, \mathcal{C}^0_3)\end{array}$$

$$\xRightarrow{\overline{c}\langle \mathsf{ax}_1\rangle}_{s} \begin{array}{c}(\mathcal{Q}_2, \mathcal{C}_2),\\(\mathcal{P}^0_4, \mathcal{C}^0_4)\\\mathbf{X = 0}\end{array}$$

$$\xRightarrow{\overline{c}\langle \mathsf{ax}_1\rangle}_{s} \begin{array}{c}(\mathcal{Q}_2, \mathcal{C}_2),\\(\mathcal{P}^0_5, \mathcal{C}^0_5)\\\mathbf{X \neq 0}\end{array}$$

Need to **partition**: $\mathcal{C}^0_4$ enforces $X = 0$ and $\mathcal{C}^0_5$ enforces $X \neq 0$.

Partition Tree

Build a **joint** symbolic execution tree

Partition solutions (split nodes): ensure static equivalences of all solutions in a same node
$\rightsquigarrow$ done by **constraint solving algorithm**

$$\begin{matrix}(\mathcal{Q}_0,\mathcal{C}_\emptyset)\\(\mathcal{P}_0^0,\mathcal{C}_\emptyset)\end{matrix} \xRightarrow{c(X)}_{s} \begin{matrix}(\mathcal{Q}_1,\mathcal{C}_1),\ (\mathcal{P}_1^0,\mathcal{C}_1^0)\\(\mathcal{P}_2^0,\mathcal{C}_2^0),\ (\mathcal{P}_3^0,\mathcal{C}_3^0)\end{matrix}$$

$$\xRightarrow{\overline{c}\langle ax_1\rangle}_{s} \begin{matrix}(\mathcal{Q}_2,\mathcal{C}_2),\\(\mathcal{P}_4^0,\mathcal{C}_4^0)\\\mathbf{X = 0}\end{matrix} \qquad \xRightarrow{\overline{c}\langle ax_1\rangle}_{s} \begin{matrix}(\mathcal{Q}_2,\mathcal{C}_2),\\(\mathcal{P}_5^0,\mathcal{C}_5^0)\\\mathbf{X \neq 0}\end{matrix}$$

Need to **partition**: $\mathcal{C}_4^0$ enforces $X = 0$ and $\mathcal{C}_5^0$ enforces $X \neq 0$.
$\mathbf{P^0 \approx_t Q}$: each leaf contains processes derived from P^0 and Q.

Partition Tree

Build a **joint** symbolic execution tree

Partition solutions (split nodes): ensure static equivalences of all solutions in a same node
$\rightsquigarrow$ done by **constraint solving algorithm**

$$\begin{tikzcd}[column sep=large]
 & & \begin{array}{c}(\mathcal{Q}_2,\mathcal{C}_2),\\ \mathbf{X=1}\end{array} \\
\begin{array}{c}(\mathcal{Q}_0,\mathcal{C}_\emptyset)\\(\mathcal{P}^1_0,\mathcal{C}_\emptyset)\end{array} \arrow[r, Rightarrow, "c(X)", "\mathsf{s}"'] & \begin{array}{c}(\mathcal{Q}_1,\mathcal{C}_1),\ (\mathcal{P}^1_1,\mathcal{C}^1_1)\\(\mathcal{P}^1_2,\mathcal{C}^1_2),\ (\mathcal{P}^1_3,\mathcal{C}^1_3)\end{array} \arrow[ur, Rightarrow, "\overline{c}\langle ax_1\rangle", "\mathsf{s}"'] \arrow[r, Rightarrow, "\overline{c}\langle ax_1\rangle", "\mathsf{s}"'] \arrow[dr, Rightarrow, "\overline{c}\langle ax_1\rangle", "\mathsf{s}"'] & \begin{array}{c}(\mathcal{P}^1_4,\mathcal{C}^1_4)\\ \mathbf{X=1}\end{array} \\
 & & \begin{array}{c}(\mathcal{Q}_2,\mathcal{C}_2),\\(\mathcal{P}^0_5,\mathcal{C}^0_5)\\ \mathbf{X\neq 1}\end{array}
\end{tikzcd}$$

Need to **partition more** to ensure static equivalence inside nodes.
$\mathbf{P^1 \not\approx_t Q}$: leaves with processes only from P^1.

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, loops, ...)	strands
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, loops, ...)	strands
eq. theories	finite variant (?)	subterm conv. + DH	finite variant + algebraic prop.

Bounded number of sessions

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

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, loops, ...)	strands
eq. theories	finite variant (?)	subterm conv. + DH	finite variant + algebraic prop.

Bounded number of sessions

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

No swiss knife for equivalence properties