Automated verification of equivalence properties in symbolic models

Steve Kremer

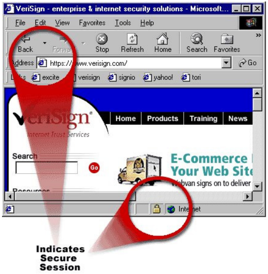
INRIA Nancy & LORIA

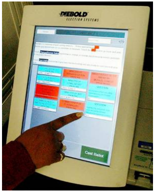
15/10/2013 - University of Luxembourg

Cryptographic protocols everywhere!

Cryptographic protocol:

a distributed program which uses cryptographic primitives (e.g. encryption, digital signatures, ...) to ensure a security property (e.g. confidentiality, authentication, anonymity, ...)





Cryptographic protocols everywhere!

Cryptographic protocol:

a distributed program which uses cryptographic primitives (e.g. encryption, digital signatures, ...) to ensure a security property (e.g. confidentiality, authentication, anonymity, ...)

FEVAD 2011 key numbers for France

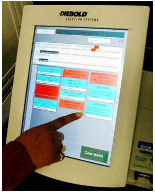
fédération du e-commerce et de la vente à distance

- 28 millions customers buying online (end 1st trimester 2011)
- online sales for an amount of 31 billions € (in 2010)

Cryptographic protocols everywhere!

Cryptographic protocol:

a distributed program which uses cryptographic primitives (e.g. encryption, digital signatures, . . .) to ensure a security property (e.g. confidentiality, authentication, anonymity, . . .)



Political Internet elections in Europe since 2011

- parlament elections in Estonia, Switzerland and recently France
- regional elections in Norway

Symbolic, automated verification of equivalence properties

Symbolic (as opposed to computational) models:

- messages are terms (labelled trees)
- computationally unbounded adversary that controls the network
- explicit rules for computing on terms (perfect cryptography assumption)

  $dec(enc(m, k, r), k) \to m$

Symbolic, automated verification of equivalence properties

Symbolic (as opposed to computational) models:

- messages are terms (labelled trees)
- computationally unbounded adversary that controls the network
- explicit rules for computing on terms (perfect cryptography assumption)

  $$dec(enc(m, k, r), k) \to m$$

Successful approach to automatically verify protocols and find flaws

- Flaw found in Single Sign On Protocols, used e.g., in Google Apps
- Attacks on commercial security tokens implementing the PKCS#11 standard

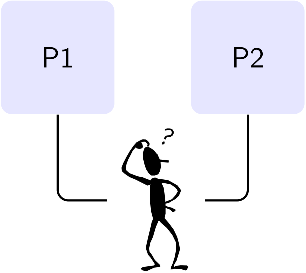
Plethora of good tools which can handle a variety of protocols:
AVISPA, Casper, Maude-NPA, ProVerif, Scyther, Tamarin, ...

Part I

The notion of indistinguishability and why it matters

Indistinguishability (informally)

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



We write $P1 \approx P2$ when the adversary cannot distinguish $P1$ and $P2$

Indistinguishability in symbolic models

In symbolic models indistinguishability is naturally modelled using equivalences from process calculi, e.g. [Spi calculus, Abadi & Gordon'96], [Applied pi calculus, Abadi & Fournet'01]

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.

Indistinguishability in symbolic models

In symbolic models indistinguishability is naturally modelled using equivalences from process calculi, e.g. [Spi calculus, Abadi & Gordon'96], [Applied pi calculus, Abadi & Fournet'01]

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.

Remarks

- Process equivalences are well known notions in concurrency theory; much more difficult when adding support for crypto primitives
- A whole zoo of equivalences (with subtle differences)

Indistinguishability in computational models

In computational models indistinguishability is the most standard notion for expressing properties!

Computational indistinguishability ($P \approx Q$)

for all PPT Turing machine $\mathcal{A}$, we have that:

$$|Pr[\mathcal{A}^P(\eta) = 1] - Pr[\mathcal{A}^Q(\eta) = 1]|$$

is negligible in the security parameter η

Conceptually the same ideas! Strangely, appeared in symbolic models much later.

Secrecy in symbolic models

In symbolic analysis secrecy is generally modelled as non-deducibility:

the attacker cannot compute the value of the secret

$\rightsquigarrow$ partial leakage is not detected

Example (Weak secrecy)

Let h be a one-way hash function. The protocol $P = \nu s.out(c, h(s))$ 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]

$$\mathrm{in}(c, \langle t_1, t_2 \rangle).\ P\{^{t_1}/_{s}\} \approx \mathrm{in}(c, \langle t_1, t_2 \rangle).\ P\{^{t_2}/_{s}\}$$

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.

- Resistance against offline guessing attacks (real-or-random): [Corin et al.'05]

$$P; \mathrm{out}(s) \approx P; \nu s'.\mathrm{out}(s')$$

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

Anonymity properties

Privacy in electronic voting: [K., Ryan'05]

$$V\{^a/_{id}\}\{^0/_v\} \mid V\{^b/_{id}\}\{^1/_v\} \approx V\{^a/_{id}\}\{^1/_v\} \mid V\{^b/_{id}\}\{^0/_v\}$$

Can also be used to model strong versions of privacy: receipt-freeness, coercion-resistance, everlasting privacy

Many others anonymity properties:

- Unlinkability in RFID protocols (e.g. electronic passports), mobile telephony, vehicular networks, ... e.g. [Arapinis et al.'10,'11,'12]
- Private authentication [Abadi,Fournet'02]
- Privacy in e-auctions [Dong et al.'10, Dreier et al.'13]
- ...

The ideal system approach

- The protocol P is indistinguishable from an ideal system (or specification) I which is secure by construction

$$P \approx I$$

Example ([Abadi, Gordon'96])

$$\begin{array}{rcl} P(m) & = & \nu k.(\nu r.\mathrm{out}(c, \mathrm{senc}(m,k,r)) \mid (\mathrm{in}(c,x).\mathrm{let}\ y = \mathrm{sdec}(x,k).F(y))) \\ I(m) & = & \nu k.(\nu r.\mathrm{out}(c, \mathrm{senc}(m,k,r)) \mid (\mathrm{in}(c,x).\mathrm{let}\ y = \mathrm{sdec}(x,k).F(m))) \end{array}$$

Authenticity: for all m. $P(m) \approx I(m)$

The ideal system approach

- The protocol P is indistinguishable from an ideal system (or specification) I which is secure by construction

$$P \approx I$$

Example ([Abadi, Gordon'96])

$$\begin{array}{rcl} P(m) & = & \nu k.(\nu r.\mathrm{out}(c, \mathrm{senc}(m,k,r)) \mid (\mathrm{in}(c,x).\mathrm{let}\ y = \mathrm{sdec}(x,k).F(y))) \\ I(m) & = & \nu k.(\nu r.\mathrm{out}(c, \mathrm{senc}(m,k,r)) \mid (\mathrm{in}(c,x).\mathrm{let}\ y = \mathrm{sdec}(x,k).F(m))) \end{array}$$

Authenticity: for all m. $P(m) \approx I(m)$

- Simulation based security [Canetti'01], [Pfitzmann, Waidner'01], ...

$$\exists S.\ P \approx S \mid I$$

First appeared in computational models, more recently in symbolic models [Delaune, K., Pereira'09],[Böhl, Unruh'13]

Automated verification?

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
- correspondence properties

  If B ended a session with parameter p then A must have started a session with parameters p'.

Automated verification?

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
- correspondence properties

If B ended a session with parameter p then A must have started a session with parameters p'.

Verifying indistinguishability properties?!?

In the above list ProVerif is the only tool able to verify (some) indistinguishability properties

Part II

Automated verification of indistinguishability properties: our approach and how we (partially) failed

Existing work on verifying equivalence properties

- NP completeness results for equivalence of two symbolic traces [Baudet'05,Chevalier & Rusinowitch'10]
  $\rightsquigarrow$ allows to verify trace equivalence for a class of simple processes for a bounded number of sessions [Cortier & Delaune'10]
  $\rightsquigarrow$ procedures are highly non-deterministic and not reasonably implementable

Existing work on verifying equivalence properties

- NP completeness results for equivalence of two symbolic traces [Baudet'05,Chevalier & Rusinowitch'10]

  $\rightsquigarrow$ allows to verify trace equivalence for a class of simple processes for a bounded number of sessions [Cortier & Delaune'10]

  $\rightsquigarrow$ procedures are highly non-deterministic and not reasonably implementable

- more practical procedures [Cheval, Comon-Lundh, Delaune '10,'11, Dawson & Tiu'10]

  $\rightsquigarrow$ restricted support of cryptographic primitives (encryption, signatures, hash)

Existing work on verifying equivalence properties

- NP completeness results for equivalence of two symbolic traces [Baudet'05,Chevalier & Rusinowitch'10]
  $\rightsquigarrow$ allows to verify trace equivalence for a class of simple processes for a bounded number of sessions [Cortier & Delaune'10]
  $\rightsquigarrow$ procedures are highly non-deterministic and not reasonably implementable

- more practical procedures [Cheval, Comon-Lundh, Delaune '10,'11, Dawson & Tiu'10]
  $\rightsquigarrow$ restricted support of cryptographic primitives (encryption, signatures, hash)

- equivalence verified by ProVerif [Blanchet, Abadi & Fournet'05]
  $\rightsquigarrow$ efficient procedure for an unbounded number of sessions, but due to approximations proofs fail for some interesting protocols
  $\rightsquigarrow$ recent results allow to check more equivalences (but still restricted) [Blanchet, Cheval'13]

Our goals and approach [Chadha, Ciobâcă, K., 2012]

Decision procedure for trace equivalence for a simple core labelled semantics :

- many equational theories,
- practical implementation

First order Horn clauses modelling of protocols for a bounded number of sessions

(as opposed to the usual modelling in Horn clauses for an unbounded number of sessions, allowing false attacks)

Resolution based procedure for trace equivalence for convergent equational theories (in particular optimally reducing eq. theories)

Terms and frames

Messages are modelled as first-order terms equipped with a convergent rewrite system R.

Secret values are modelled as names in a set $\mathcal{N}$.

We write $t =_{\mathsf{R}} u$ when $t\downarrow = u\downarrow$

Example

Signature: senc/3, sdec/2, pair/2, fst/1, snd/1, 0/0, 1/0

Rewrite system:
$\mathsf{sdec}(\mathsf{senc}(x, y, z), y) \to_{\mathsf{R}} x, \mathsf{fst}(\mathsf{pair}(x, y)) \to_{\mathsf{R}} x, \mathsf{snd}(pair(x, y)) \to_{\mathsf{R}} y$

Terms: $t_1 = \mathsf{senc}(n, k, r)$, $t_2 = \mathsf{sdec}(t_1, k) \quad (n, k, r \in \mathcal{N})$
We have that $t_2 =_{\mathsf{R}} n$

Sequences of messages are grouped in a frame $\varphi = \{^{t_1}/_{w_1}, \ldots, ^{t_n}/_{w_n}\}$

Deduction

What messages can an attacker compute?

Definition (Deduction)

A term t is *deducible from frame* φ *with a recipe* r ($\varphi \vdash^r t$) if $r\varphi =_{\mathsf{R}} t$ and r does not contain names in $\mathcal{N}$.

Example

Let $\varphi = \{^{\mathsf{senc}(n_1,k_1,r_1)}/_{w_1}, ^{\mathsf{senc}(n_2,k_2,r_2)}/_{w_2}, ^{k_1}/_{w_3}\}$.

We have that $\varphi \vdash^{\mathsf{sdec}(w_1,w_3)} n_1$, $\varphi \not\vdash n_2$, $\varphi \vdash^1 1$

Static equivalence

Indistinguishability of sequences of messages

Definition (Static equivalence)

$(r_1 = r_2)\varphi$ if $\varphi \vdash^{r_1} t$ and $\varphi \vdash^{r_2} t$ for some t.

φ_1 *statically equivalent* to φ_2 ($\varphi_1 \approx_s \varphi_2$) iff $(r_1 = r_2)\varphi_1 \Leftrightarrow (r_1 = r_2)\varphi_2$.

Examples

$$\{{}^{n_1}/_{w_1}\} \approx_s \{{}^{n_2}/_{w_1}\}$$

$$\{{}^{n_1}/_{w_1}, {}^{n_2}/_{w_2}\} \not\approx_s \{{}^{n_1}/_{w_1}, {}^{n_1}/_{w_2}\} \qquad (w_1 \stackrel{?}{=} w_2)$$

$$\{{}^{\mathrm{senc}(0,k,r)}/_{w_1}\} \approx_s \{{}^{\mathrm{senc}(1,k,r)}/_{w_1}\}$$

$$\{{}^{\mathrm{senc}(n,k,r)}/_{w_1}, {}^{k}/_{w_2}\} \not\approx_s \{{}^{\mathrm{senc}(0,k,r)}/_{w_1}, {}^{k}/_{w_2}\} \qquad (sdec(w_1, w_2) \stackrel{?}{=} 0)$$

A simple crypto process calculus: syntax

Actions : $\mathsf{in}(c, x) \mid \mathsf{out}(c, t) \mid [s \stackrel{?}{=} t]$

Symbolic Trace: sequence of actions

Example

$$\begin{array}{rcl} T & = & \mathsf{out}(c, \mathsf{enc}(a, k)).\mathsf{out}(c, \mathsf{enc}(a', k)). \\ & & \mathsf{in}(c, x).\mathsf{out}(c, \mathsf{dec}(x, k)). \\ & & \mathsf{in}(c, y).[y \stackrel{?}{=} \mathsf{pair}(a, a')].\mathsf{out}(c, s) \end{array}$$

Process: set of symbolic traces

Remark: Parallel composition $(P \mid Q)$ can be defined as the set of interleavings

A simple crypto process calculus: semantics

Operational semantics: $(T, \varphi) \xrightarrow{\ell} (T', \varphi')$

$$\text{Receive}\ \frac{\varphi \vdash^{r} t}{(\mathrm{in}(c,x).T, \varphi) \xrightarrow{\mathrm{in}(c,r)} (T\{x \mapsto t\}, \varphi)}$$

$$\text{Test}\ \frac{s =_{\mathrm{R}} t}{([s \stackrel{?}{=} t].T, \varphi) \xrightarrow{\mathrm{test}} (T, \varphi)}$$

$$\text{Send}\ \frac{}{(\mathrm{out}(c,t).T, \varphi) \xrightarrow{\mathrm{out}(c)} (T, \varphi \cup \{w_{|dom(\varphi)|+1} \mapsto t\})}$$

$P \xrightarrow{\ell} (T', \varphi)$ if $\exists T \in P.\ (T, \emptyset) \xrightarrow{\ell} (T', \varphi)$

$\xRightarrow{\ell}$ if $\xrightarrow{\mathrm{test}^*\ \ell\ \mathrm{test}^*}$: weak semantics hiding silent test actions

Trace equivalences

Trace equivalence

$P \sqsubseteq_t Q$ if $(P, \emptyset) \xrightarrow{\ell_1,\ldots,\ell_n} (P', \varphi)$ implies $\exists Q', \varphi'.\ (Q, \emptyset) \xrightarrow{\ell_1,\ldots,\ell_n} (Q', \varphi') \wedge \varphi \sim_s \varphi'$.

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

Trace equivalences

Fine grained trace equivalence

$P \sqsubseteq_{ft} Q$ if $\forall T \in P.\ \exists T' \in Q.\ T \approx_t T'$

Trace equivalence

$P \sqsubseteq_t Q$ if $(P, \emptyset) \xrightarrow{\ell_1, \ldots, \ell_n} (P', \varphi)$ implies $\exists Q', \varphi'.\ (Q, \emptyset) \xrightarrow{\ell_1, \ldots, \ell_n} (Q', \varphi') \wedge \varphi \sim_s \varphi'$.

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

Trace equivalences

Fine grained trace equivalence

$P \sqsubseteq_{ft} Q$ if $\forall T \in P.\ \exists T' \in Q.\ T \approx_t T'$

$\Downarrow \qquad \not\Uparrow$

Trace equivalence

$P \sqsubseteq_t Q$ if $(P, \emptyset) \xRightarrow{\ell_1,\ldots,\ell_n} (P', \varphi)$ implies $\exists Q', \varphi'.\ (Q, \emptyset) \xRightarrow{\ell_1,\ldots,\ell_n} (Q', \varphi') \wedge \varphi \sim_s \varphi'$.

Example:

$P \approx_t Q$ but $P \not\approx_{ft} Q$

$$P = \{\mathsf{out}(c, enc(a,k)).\mathsf{out}(c, enc(b,k)).\mathsf{in}(c,x).[x = enc(a,k)].\mathsf{out}(c,k),\ \mathsf{out}(c, enc(a,k)).\mathsf{out}(c, enc(b,k)).\mathsf{in}(c,x).[x = enc(b,k)].\mathsf{out}(c,k)\}$$

$$Q = \{\mathsf{out}(c, enc(a,k)).\mathsf{out}(c, enc(b,k)).\mathsf{in}(c,x).[x = enc(dec(x,k),k)].\mathsf{out}(c,k)\}$$

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

Trace equivalences

Fine grained trace equivalence

$P \sqsubseteq_{ft} Q$ if $\forall T \in P.\ \exists T' \in Q.\ T \approx_t T'$

$\Downarrow \qquad \not\Uparrow$

Trace equivalence

$P \sqsubseteq_t Q$ if $(P, \emptyset) \xRightarrow{\ell_1,\ldots,\ell_n} (P', \varphi)$ implies $\exists Q', \varphi'.\ (Q, \emptyset) \xRightarrow{\ell_1,\ldots,\ell_n} (Q', \varphi') \wedge \varphi \sim_s \varphi'$.

Coarse trace equivalence

$P \sqsubseteq_{ct} Q$ if $(P, \emptyset) \xRightarrow{\ell_1,\ldots,\ell_n} (P', \varphi) \wedge (r = s)\varphi$ implies $\exists Q', \varphi'.\ (Q, \emptyset) \xRightarrow{\ell_1,\ldots,\ell_n} (Q', \varphi') \wedge (r = s)\varphi'$

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

Trace equivalences

Fine grained trace equivalence

$P \sqsubseteq_{ft} Q$ if $\forall T \in P.\ \exists T' \in Q.\ T \approx_t T'$

$\Downarrow \qquad \not\Uparrow$

Trace equivalence

$P \sqsubseteq_t Q$ if $(P, \emptyset) \xRightarrow{\ell_1, \ldots, \ell_n} (P', \varphi)$ implies $\exists Q', \varphi'.\ (Q, \emptyset) \xRightarrow{\ell_1, \ldots, \ell_n} (Q', \varphi') \wedge \varphi \sim_s \varphi'$.

$\Downarrow \qquad \not\Uparrow$

Coarse trace equivalence

$P \sqsubseteq_{ct} Q$ if $(P, \emptyset) \xRightarrow{\ell_1, \ldots, \ell_n} (P', \varphi) \wedge (r = s)\varphi$ implies $\exists Q', \varphi'.\ (Q, \emptyset) \xRightarrow{\ell_1, \ldots, \ell_n} (Q', \varphi') \wedge (r = s)\varphi'$

Example:

$P \approx_{ct} Q$ but $P \not\approx_t Q$

$P = \{\mathrm{out}(c, a).\mathrm{out}(c, a)\}$

$Q = \{\mathrm{out}(c, a).\mathrm{out}(c, a),$
$\quad \mathrm{out}(c, a).\mathrm{out}(c, b)\}.$

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

Trace equivalences

Fine grained trace equivalence

$P \sqsubseteq_{ft} Q$ if $\forall T \in P.\ \exists T' \in Q.\ T \approx_t T'$

Trace equivalence

$P \sqsubseteq_t Q$ if $(P, \emptyset) \xrightarrow{\ell_1,\dots,\ell_n} (P', \varphi)$ implies $\exists Q', \varphi'.\ (Q, \emptyset) \xrightarrow{\ell_1,\dots,\ell_n} (Q', \varphi') \wedge \varphi \sim_s \varphi'$.

det. proc.

Coarse trace equivalence

$P \sqsubseteq_{ct} Q$ if $(P, \emptyset) \xrightarrow{\ell_1,\dots,\ell_n} (P', \varphi) \wedge (r = s)\varphi$ implies $\exists Q', \varphi'.\ (Q, \emptyset) \xrightarrow{\ell_1,\dots,\ell_n} (Q', \varphi') \wedge (r = s)\varphi'$

Definition:

P is *determinate* if whenever $(P, \emptyset) \xrightarrow{\ell_1,\dots,\ell_n} (T, \varphi)$ and $(P, \emptyset) \xrightarrow{\ell_1,\dots,\ell_n} (T', \varphi')$ then $\varphi \approx_s \varphi'$.

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

Trace equivalences

Fine grained trace equivalence

$P \sqsubseteq_{ft} Q$ if $\forall T \in P.\ \exists T' \in Q.\ T \approx_t T'$

$\Downarrow \qquad \not\Uparrow$

Trace equivalence

$P \sqsubseteq_t Q$ if $(P, \emptyset) \xRightarrow{\ell_1, \ldots, \ell_n} (P', \varphi)$ implies $\exists Q', \varphi'.\ (Q, \emptyset) \xRightarrow{\ell_1, \ldots, \ell_n} (Q', \varphi') \wedge \varphi \sim_s \varphi'$.

$\Downarrow \qquad \not\Uparrow \qquad \Uparrow$ det. proc.

Coarse trace equivalence

$P \sqsubseteq_{ct} Q$ if $(P, \emptyset) \xRightarrow{\ell_1, \ldots, \ell_n} (P', \varphi) \wedge (r = s)\varphi$ implies $\exists Q', \varphi'.\ (Q, \emptyset) \xRightarrow{\ell_1, \ldots, \ell_n} (Q', \varphi') \wedge (r = s)\varphi'$

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

Remark:

A trace is a determinate process

Definition:

P is *determinate* if whenever $(P, \emptyset) \xRightarrow{\ell_1, \ldots, \ell_n} (T, \varphi)$ and $(P, \emptyset) \xRightarrow{\ell_1, \ldots, \ell_n} (T', \varphi')$ then $\varphi \approx_s \varphi'$.

Our procedure: overview

1. Model protocol and intruder capabilities in Horn clauses
2. Saturate clauses using dedicated resolution procedure
3. Check equivalence

Our procedure: overview

1. Model protocol and intruder capabilities in Horn clauses
2. Saturate clauses using dedicated resolution procedure
3. Check equivalence

We fail to verify trace equivalence (in general) :-(

- under-approximate trace equivalence ($\approx_{ft}$)
- over-approximate trace equivalence ($\approx_{ct}$)
- verify trace equivalence for determinate processes

1. Horn clause modelling

$$T \;=\; \mathsf{out}(c, \mathsf{enc}(a,k)).\mathsf{out}(c,\mathsf{enc}(a',k)).\mathsf{in}(c,x).\mathsf{out}(c,\mathsf{dec}(x,k)).$$
$$\mathsf{in}(c,y).[y \stackrel{?}{=} \mathsf{pair}(a,a')].\mathsf{out}(c,s)$$

$k(R,t)$: attacker knowledge predicate (*attacker can compute t using recipe R*)

Compute a initial set for trace T: seed(T)

$$\begin{array}{l}
\mathsf{k}(w_1, \mathsf{enc}(a,k)) \\
\mathsf{k}(w_2, \mathsf{enc}(a',k)) \\
\mathsf{k}(w_3, \mathsf{dec}(x,k)) \Leftarrow \mathsf{k}(X,x) \\
\mathsf{k}(w_4, s) \Leftarrow \mathsf{k}(X,x), \mathsf{k}(Y,y), y =_{\mathsf{R}} \mathsf{pair}(a,a')
\end{array}$$

1. Horn clause modelling

$$T = \mathsf{out}(c, \mathsf{enc}(a, k)).\mathsf{out}(c, \mathsf{enc}(a', k)).\mathsf{in}(c, x).\mathsf{out}(c, \mathsf{dec}(x, k)).$$
$$\mathsf{in}(c, y).[y \stackrel{?}{=} \mathsf{pair}(a, a')].\mathsf{out}(c, s)$$

$k(R, t)$: attacker knowledge predicate (*attacker can compute t using recipe R*)

Compute a initial set for trace T: seed(T)

$$\begin{array}{l}
\mathsf{k}(w_1, \mathsf{enc}(a, k)) \\
\mathsf{k}(w_2, \mathsf{enc}(a', k)) \\
\mathsf{k}(w_3, \mathsf{dec}(x, k)) \Leftarrow \mathsf{k}(X, x) \\
\mathsf{k}(w_4, s) \Leftarrow \mathsf{k}(X, x), \mathsf{k}(Y, y), y =_{\mathsf{R}} \mathsf{pair}(a, a')
\end{array}$$

1. Horn clause modelling

$$T = \mathsf{out}(c, \mathsf{enc}(a, k)).\mathsf{out}(c, \mathsf{enc}(a', k)).\mathsf{in}(c, x).\mathsf{out}(c, \mathsf{dec}(x, k)).$$
$$\mathsf{in}(c, y).[y \stackrel{?}{=} \mathsf{pair}(a, a')].\mathsf{out}(c, s)$$

$k(R, t)$: attacker knowledge predicate (*attacker can compute t using recipe R*)

Compute a initial set for trace T: seed(T)

$$\mathsf{k}(w_1, \mathsf{enc}(a, k))$$
$$\mathsf{k}(w_2, \mathsf{enc}(a', k))$$
$$\mathsf{k}(w_3, \mathsf{dec}(x, k)) \Leftarrow \mathsf{k}(X, x)$$
$$\mathsf{k}(w_4, s) \Leftarrow \mathsf{k}(X, x), \mathsf{k}(Y, y), y =_{\mathsf{R}} \mathsf{pair}(a, a')$$

1. Horn clause modelling

$$T = \mathrm{out}(c, \mathrm{enc}(a,k)).\mathrm{out}(c, \mathrm{enc}(a',k)).\mathrm{in}(c,x).\mathrm{out}(c, \mathrm{dec}(x,k)). \\ \mathrm{in}(c,y).[y \stackrel{?}{=} \mathrm{pair}(a,a')].\mathrm{out}(c,s)$$

$k(R,t)$: attacker knowledge predicate (*attacker can compute t using recipe R*)

Compute a initial set for trace T: seed(T)

$$\begin{array}{l} \mathrm{k}(w_1, \mathrm{enc}(a,k)) \\ \mathrm{k}(w_2, \mathrm{enc}(a',k)) \\ \mathrm{k}(w_3, \mathrm{dec}(x,k)) \Leftarrow \mathrm{k}(X,x) \\ \mathrm{k}(w_4, s) \Leftarrow \mathrm{k}(X,x), \mathrm{k}(Y,y), y =_{\mathrm{R}} \mathrm{pair}(a,a') \end{array}$$

1. Horn clause modelling

$$
\begin{aligned}
T \quad = \quad & \mathsf{out}(c, \mathsf{enc}(a, k)).\mathsf{out}(c, \mathsf{enc}(a', k)).\mathsf{in}(c, x).\mathsf{out}(c, \mathsf{dec}(x, k)). \\
& \mathsf{in}(c, y).[y \stackrel{?}{=} \mathsf{pair}(a, a')].\mathsf{out}(c, s)
\end{aligned}
$$

$k(R, t)$: attacker knowledge predicate (*attacker can compute t using recipe R*)

Compute a initial set for trace T: seed(T)

$$
\begin{aligned}
& \mathsf{k}(w_1, \mathsf{enc}(a, k)) \\
& \mathsf{k}(w_2, \mathsf{enc}(a', k)) \\
& \mathsf{k}(w_3, \mathsf{dec}(x, k)) \Leftarrow \mathsf{k}(X, x) \\
& \mathsf{k}(w_4, s) \Leftarrow \mathsf{k}(X, x), \mathsf{k}(Y, y), y =_{\mathsf{R}} \mathsf{pair}(a, a')
\end{aligned}
$$

1. Horn clause modelling

$$T = \mathsf{out}(c, \mathsf{enc}(a, k)).\mathsf{out}(c, \mathsf{enc}(a', k)).\mathsf{in}(c, x).\mathsf{out}(c, \mathsf{dec}(x, k)).$$
$$\mathsf{in}(c, y).[y \stackrel{?}{=} \mathsf{pair}(a, a')].\mathsf{out}(c, s)$$

$k(R, t)$: attacker knowledge predicate (*attacker can compute t using recipe R*)

Compute a initial set for trace T: seed(T)

$$\mathsf{k}_{\mathsf{out}(c)}(w_1, \mathsf{enc}(a, k))$$
$$\mathsf{k}_{\mathsf{out}(c),\mathsf{out}(c)}(w_2, \mathsf{enc}(a', k))$$
$$\mathsf{k}_{\mathsf{out}(c),\mathsf{out}(c),\mathsf{in}(c,x),\mathsf{out}(c)}(w_3, \mathsf{dec}(x, k)) \Leftarrow \mathsf{k}_{\mathsf{out}(c),\mathsf{out}(c)}(X, x)$$
$$\mathsf{k}_{\mathsf{out}(c),\mathsf{out}(c),\mathsf{in}(c,x),\mathsf{out}(c),\mathsf{in}(c,y),\mathsf{out}(c)}(w_4, s) \Leftarrow$$
$$\mathsf{k}_{\mathsf{out}(c),\mathsf{out}(c)}(X, x), \mathsf{k}_{\mathsf{out}(c),\mathsf{out}(c),\mathsf{in}(c,x),\mathsf{out}(c)}(Y, y), y =_{\mathsf{R}} \mathsf{pair}(a, a').$$

Add history for accuracy (avoid false attacks)

1. Horn clause modelling

$$T \quad = \quad \mathrm{out}(c, \mathrm{enc}(a, k)).\mathrm{out}(c, \mathrm{enc}(a', k)).\mathrm{in}(c, x).\mathrm{out}(c, \mathrm{dec}(x, k)).$$
$$\mathrm{in}(c, y).[y \stackrel{?}{=} \mathrm{pair}(a, a')].\mathrm{out}(c, s)$$

$k(R, t)$: attacker knowledge predicate (*attacker can compute t using recipe R*)

Compute a initial set for trace T: seed(T)

$$\mathrm{k}_{\mathrm{out}(c)}(w_1, \mathrm{enc}(a, k))$$
$$\mathrm{k}_{\mathrm{out}(c),\mathrm{out}(c)}(w_2, \mathrm{enc}(a', k))$$
$$\mathrm{k}_{\mathrm{out}(c),\mathrm{out}(c),\mathrm{in}(c,x),\mathrm{out}(c)}(w_3, \mathrm{dec}(x, k)) \Leftarrow \mathrm{k}_{\mathrm{out}(c),\mathrm{out}(c)}(X, x)$$
$$\mathrm{k}_{\mathrm{out}(c),\mathrm{out}(c),\mathrm{in}(c,x),\mathrm{out}(c),\mathrm{in}(c,y),\mathrm{out}(c)}(w_4, s) \Leftarrow$$
$$\mathrm{k}_{\mathrm{out}(c),\mathrm{out}(c)}(X, x), \mathrm{k}_{\mathrm{out}(c),\mathrm{out}(c),\mathrm{in}(c,x),\mathrm{out}(c)}(Y, y), y =_{\mathrm{R}} \mathrm{pair}(a, a').$$

Add history for accuracy (avoid false attacks)

Clauses for attacker capabilities:
$\mathrm{k}_w(f(X_1, \ldots X_n), f(x_1, \ldots, x_k)) \Leftarrow \mathrm{k}_w(X_1, x_1), \ldots, \mathrm{k}_w(X_k, x_k)$

1. Horn clause modelling: getting rid of equations

Use equational unification to remove tests:

$$\Big(H \Leftarrow B_1, \ldots, B_n, u =_{\mathsf{R}} v\Big) \quad \rightsquigarrow \quad \begin{array}{l} \Big((H \Leftarrow B_1, \ldots, B_n)\sigma_1\Big) \\ \ldots \\ \Big((H \Leftarrow B_1, \ldots, B_n)\sigma_k\Big) \end{array}$$

where $\sigma_1, \ldots, \sigma_k$ is a complete set of unifiers for $u =_{\mathsf{R}} v$.

1. Horn clause modelling: getting rid of equations (2)

Use finite variant property ([Comon-Lund, Delaune'05]) to get rid of equational reasoning:

> *Finite variant property: possibility to precompute a finite set of all possible normal forms*

$$\Big(\mathsf{k}_h(R,t) \Leftarrow B_1,\ldots,B_n\Big) \quad \rightsquigarrow \quad \begin{array}{l} \Big((\mathsf{k}_H(R,t))\theta_1{\downarrow} \Leftarrow B_1\theta_1{\downarrow},\ldots,B_n\theta_1{\downarrow}\Big) \\ \ldots \\ \Big((\mathsf{k}_H(R,t))\theta_k{\downarrow} \Leftarrow B_1\theta_k{\downarrow},\ldots,B_n\theta_k{\downarrow}\Big). \end{array}$$

where $\theta_1,\ldots,\theta_k$ is a complete set of variants for t.

We can compute finite sets of variants and mgu_E for the class of optimally reducing theories (contains subterm convergent, blind sigs, td commitment, ...)

1. Horn clause modelling: predicates

Predicates: interpreted over ground trace

- Reachability predicate

  $T \models \mathsf{r}_{\ell_1,\ldots,\ell_n}$ if $(T,\emptyset) \xrightarrow{L_1} (T_1,\varphi_1) \xrightarrow{L_2} \ldots \xrightarrow{L_n} (T_n,\varphi_n)$ such that $\ell_i =_{\mathsf{R}} L_i\varphi_{i-1}$ for all $1 \leq i \leq n$

- intruder Knowledge predicate

  $T \models \mathsf{k}_{\ell_1,\ldots,\ell_n}(R,t)$ if when $\mathsf{r}_{\ell_1,\ldots,\ell_n}$ then $\varphi_n \vdash^{R\sigma} t\sigma$

- Identity predicate

  $T \models \mathsf{i}_{\ell_1,\ldots,\ell_n}(R,R')$ if $\exists t.\ T \models \mathsf{k}_{\ell_1,\ldots,\ell_i}(R,t)$ and $T \models \mathsf{k}_{\ell_1,\ldots,\ell_i}(R',t)$

- reachable identity predicates

  $T \models \mathsf{ri}_{\ell_1,\ldots,\ell_n}(R,R')$ if $T \models \mathsf{i}_{\ell_1,\ldots,\ell_n}(R,R')$ and $T \models \mathsf{r}_{\ell_1,\ldots,\ell_n}$

1. Horn clause modelling: correctness

$\mathcal{H}(K)$: least Herbrand model of the set of Horn clauses K.

Theorem (Correctness of Horn clause modelling)

Let T be a ground trace.

- *(Soundness.) For any $f \in \mathsf{seed}(T) \cup \mathcal{H}(\mathsf{seed}(T))$ we have that $T \models f$.*
- *(Completeness.) If $(T, \emptyset) \xrightarrow{L_1,\dots,L_m} (S, \varphi)$ then*
  1. $\mathsf{r}_{L_1,\dots,L_m} \in \mathcal{H}(\mathsf{seed}(T))$, *and*
  2. *if $\varphi \vdash^R t$ then $\mathsf{k}_{L_1,\dots,L_m}(R, t\!\downarrow) \in \mathcal{H}(\mathsf{seed}(T))$.*

2. Saturation: goals of saturation

Aims of saturation

- completeness of identity predicates
- completeness for solved clauses

  A clause is called solved if it is of the form
  $H \Leftarrow \mathsf{k}(X_1, x_1), \ldots, \mathsf{k}(X_k, x_k)$.

  For a set of solved clauses K checking $f \in \mathcal{H}(K)$ is easy
  (simple recursive algorithm)
  $\rightsquigarrow$ needed for checking equivalence

2. Saturation rules

Saturate seed knowledge base using the following rules

$$\text{Resolution}\ \frac{\begin{array}{c} f \in K, g \in K_{\text{solved}}, \qquad f = \left(H \Leftarrow \mathsf{k}_{uv}(X,t), B_1, \ldots, B_n\right) \\ g = \left(\mathsf{k}_w(R,t') \Leftarrow B_{n+1}, \ldots, B_m\right) \\ \sigma = \mathrm{mgu}(\mathsf{k}_u(X,t), \mathsf{k}_w(R,t')) \qquad t \notin \mathcal{X} \end{array}}{K := K \cup \left((H \Leftarrow B_1, \ldots, B_m)\sigma\right)}$$

$$\text{Equation}\ \frac{\begin{array}{c} f, g \in K_{\mathbf{solved}}, \qquad f = \left(\mathsf{k}_u(R,t) \Leftarrow B_{\mathbf{1}}, \ldots, B_n\right) \\ g = \left(\mathsf{k}_{u'v'}(R',t') \Leftarrow B_{n+\mathbf{1}}, \ldots, B_m\right) \qquad \sigma = \mathbf{mgu}(\mathsf{k}_u(\_,t), \mathsf{k}_{u'}(\_,t')) \end{array}}{K = K \cup \left((\mathsf{i}_{u'v'}(R,R') \Leftarrow B_{\mathbf{1}}, \ldots, B_m)\sigma\right)}$$

$$\text{Test}\ \frac{\begin{array}{c} f, g \in K_{\mathbf{solved}}, \\ f = \left(\mathsf{i}_u(R,R') \Leftarrow B_{\mathbf{1}}, \ldots, B_n\right) \qquad g = \left(\mathsf{r}_{u'v'} \Leftarrow B_{n+\mathbf{1}}, \ldots, B_m\right) \qquad \sigma = \mathbf{mgu}(u, u') \end{array}}{K = K \cup \left((\mathsf{ri}_{u'v'}(R,R') \Leftarrow B_{\mathbf{1}}, \ldots, B_m)\sigma\right)}$$

2. Saturation rules: soundness, completeness, termination

- Sound: If $f \in \mathrm{sat}(\mathrm{seed}(T))$ then $T \models f$

2. Saturation rules: soundness, completeness, termination

- Sound: If $f \in \mathrm{sat}(\mathrm{seed}(T))$ then $T \models f$
- Complete: If $(T, \emptyset) \xrightarrow{L_1,\ldots,L_n} (S, \varphi)$ and $K = \mathrm{sat}(\mathrm{seed}(T))$ then
  1. $\mathrm{r}_{L_1,\ldots,L_n} \in \mathcal{H}_\mathrm{e}(K_\mathrm{solved})$
  2. if $\varphi \vdash^R t$ then $\mathrm{k}_{L_1,\ldots,L_n}(R, t\!\downarrow) \in \mathcal{H}_\mathrm{e}(K_\mathrm{solved})$
  3. if $\varphi \vdash^R t$ and $\varphi \vdash^{R'} t$, then $\mathrm{i}_{L_1,\ldots,L_n}(R, R') \in \mathcal{H}_\mathrm{e}(K_\mathrm{solved})$

  where $\mathcal{H}_\mathrm{e}(K)$ the smallest set of ground terms such that
  - $\mathcal{H}(K) \subseteq \mathcal{H}_\mathrm{e}(K)$,
  - $\mathcal{H}_\mathrm{e}(K)$ is closed under congruence rules for each $\mathrm{i}_w(R, R') \in \mathcal{H}_\mathrm{e}(K)$,
  - i_w is monotonic in w.

2. Saturation rules: soundness, completeness, termination

- Sound: If $f \in \mathsf{sat}(\mathsf{seed}(T))$ then $T \models f$
- Complete: If $(T, \emptyset) \xrightarrow{L_1,\ldots,L_n} (S, \varphi)$ and $K = \mathsf{sat}(\mathsf{seed}(T))$ then
  1. $\mathsf{r}_{L_1,\ldots,L_n} \in \mathcal{H}_\mathsf{e}(K_{\mathsf{solved}})$
  2. if $\varphi \vdash^R t$ then $\mathsf{k}_{L_1,\ldots,L_n}(R, t\!\downarrow) \in \mathcal{H}_\mathsf{e}(K_{\mathsf{solved}})$
  3. if $\varphi \vdash^R t$ and $\varphi \vdash^{R'} t$, then $\mathsf{i}_{L_1,\ldots,L_n}(R, R') \in \mathcal{H}_\mathsf{e}(K_{\mathsf{solved}})$

  where $\mathcal{H}_\mathsf{e}(K)$ the smallest set of ground terms such that
  - $\mathcal{H}(K) \subseteq \mathcal{H}_\mathsf{e}(K)$,
  - $\mathcal{H}_\mathsf{e}(K)$ is closed under congruence rules for each $\mathsf{i}_w(R, R') \in \mathcal{H}_\mathsf{e}(K)$,
  - i_w is monotonic in w.
- Termination: failed to prove it :-(

  Conjectured for subterm convergent equational theories. Prototype implementation provides empirical evidence.

3. Checking equivalence

To check that $T \sqsubseteq_{ct} Q$

1. saturate: let $K = \mathrm{sat}(\mathrm{seed}(T))_{\mathrm{solved}}$
2. check reachability:
   for each $\mathrm{r}_{L_1,\ldots,L_n} \Leftarrow \mathrm{k}_{h_1}(X_1, x_1), \ldots \mathrm{k}_{h_k}(X_k, x_k) \in K$
   check that $Q, \emptyset \xrightarrow{L_1,\ldots,L_n} Q', \varphi$
3. check equalities:
   for each $\mathrm{ri}_{L_1,\ldots,L_n}(R_1, R_2) \Leftarrow \mathrm{k}_{h_1}(X_1, x_1), \ldots \mathrm{k}_{h_k}(X_k, x_k) \in K$
   check that $Q, \emptyset \xrightarrow{L_1,\ldots,L_n} Q', \varphi$ and $(R_1 = R_2)\varphi$

Tool and examples

AKiSs

(Active Knowledge In Security protocolS)

`https://github.com/ciobaca/akiss`

$\sim$2800 lines of OCaml code
(including code for computing finite variants and equational unification)

Examples:

- Strong secrecy
  NSL protocol and Blanchet's variant's of Denning-Sacco (det. processes)
- Resistance to offline guessing attacks
  EKE (det. process)
- Vote privacy FOO and Okamoto electronic voting protocols
  first completely automated proof (non-det. processes $\rightsquigarrow$ proof of $\approx_{ft}$)
- Everlasting privacy simplified versions of Helios and Moran-Naor

Part III

Future directions

Future work on AKiSs

- Termination? "Real" trace equivalence? (maybe not so important...)
- Add support for else branches
  Important: attack on e-passport relies on different error messages
  $\rightsquigarrow$ adapt techniques used in the ProVerif tool?
- Check bisimulation instead of trace equivalence
  $\rightsquigarrow$ avoid explosion of the number of traces due to interleavings
- Support for XOR and DH   wip with Baelde, Delaune
  Idea: Compute variants modulo AC and do resolution modulo AC
  $\rightsquigarrow$ Soundness/completeness (more or less) straightforward
  $\rightsquigarrow$ difficulty is termination (on practical examples): need ordered resolution

Decidability and complexity?

Unbounded number of sessions

- undecidable in general
- decidable (very restricted) fragment: ping pong protocols [CCD'13]
  ⇝ reduction to language equivalence of DPA

Bounded number of sessions

- equivalence of 2 symbolic traces
  - co-NP-complete: subterm convergent theories, no else branches [Baudet'05],[CR'10]
  - PTIME for (pure) XOR, Abelian group (with a homomorphic symbol) [DKP'12]

  ⇝ trace/observational equivalence for determinate processes [CD'09]
  ⇝ symbolic bisimulation [DKR'07]
- trace equivalence (with else branches): decidable for fixed set of crypto primitives [CCD'11]

Decidability and complexity?

Open questions

- Decidability of trace equivalence/bisimulation for other equational theories (subterm convergent, XOR, DH, . . .)?
- Complexity of trace equivalence/bisimulation?

Decidable subclasses vis tagging?

- Can we use session tags to go from bounded number of sessions to unbounded number of sessions? ($\sim$ joint state)

Not completely automated approaches?

- Verify equivalence properties using type systems
  - $\rightsquigarrow$ typing tends to be very efficient
  - $\rightsquigarrow$ some equational are difficult in automated reasoning (e.g. AC), but trivial for SMT solvers
- Allow user guidance to help tools?
  - $\rightsquigarrow$ interactive modes have been very successful, e.g. in Scyther and tamarin

Thank you for your attention!

Questions?