A procedure for verifying equivalence-based properties of cryptographic protocols

Rohit Chadha[1] Ștefan Ciobâcă[1] Steve Kremer[2]

[1] LSV & INRIA Saclay [2] LORIA & INRIA Nancy

Workshop on Computer-Aided Security

Symbolic, automated verification of equivalence properties

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:

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

Most efforts have gone into trace properties, e.g., secrecy and authentication

$\rightsquigarrow$ situation is different for indistinguishability properties

Indistinguishability based properties

Indistinguishability is extremely useful to model security properties

- Strong secrecy of s:

$$\forall t_1, t_2.\ P\{^{t_1}/_{s}\} \approx P\{^{t_2}/_{s}\}$$

- Resistance against offline guessing attacks (real-or-random):

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

- Privacy in electronic voting:

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

- and many others: indistinguishability from ideal systems (simulation based security), unlinkability in RFID systems, ...

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 interesting protocols

Indistinguishability as equivalence of processes

Indistinguishability is 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.

In many calculi such testing equivalence coincides with trace equivalence.

Our goals and approach

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 $t =_R u$ when $t{\downarrow} = u{\downarrow}$

Example

Signature: senc/3, sdec/2, pair/2, fst/1, snd/1, $\mathbf{0}$/0, $\mathbf{1}$/0
Rewrite system:
$\mathsf{sdec}(\mathsf{senc}(x, y, z), y) \to_R x, \mathsf{fst}(\mathsf{pair}(x, y)) \to_R x, \mathsf{snd}(\mathit{pair}(x, y)) \to_R y$
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 $t_2 =_R n$

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 $r\varphi =_R t$ 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 $\varphi \vdash^{\mathsf{sdec}w_1,w_3} n_1$, $\varphi \not\vdash n_2$, $\varphi \vdash^{\mathbf{1}} \mathbf{1}$

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 included* in φ_2 ($\varphi_1 \sqsubseteq_s \varphi_2$) iff $(r_1 = r_2)\varphi_1$ implies $(r_1 = r_2)\varphi_2$.

φ_1 and φ_2 are *statically equivalent* ($\varphi_1 \approx_s \varphi_2$) iff $\varphi_1 \sqsubseteq_s \varphi_2$ and $\varphi_2 \sqsubseteq_s \varphi_1$.

Examples

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

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

$$\left\{ {}^{n_1}/_{w_1}, {}^{n_2}/_{w_2} \right\} \sqsubseteq_s \left\{ {}^{n_1}/_{w_1}, {}^{n_1}/_{w_2} \right\}$$

$$\left\{ {}^{\mathsf{senc}(\mathbf{0},k,r)}/_{w_1} \right\} \approx_s \left\{ {}^{\mathsf{senc}(\mathbf{1},k,r)}/_{w_1} \right\}$$

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

A simple crypto process calculus

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

Trace: sequence of actions

Process: set of traces

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

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

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

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

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

$\stackrel{\ell}{\Longrightarrow}$ if $\xrightarrow{\mathsf{test}^* \ \ell \ \mathsf{test}^*}$: weak semantics hiding silent test actions

Trace equivalences

Trace equivalence

$P \sqsubseteq_t Q$ if $(P, \emptyset) \stackrel{\ell_1,\ldots,\ell_n}{\Longrightarrow} (P', \varphi)$ implies

$\exists Q', \varphi'.\ (Q, \emptyset) \stackrel{\ell_1,\ldots,\ell_n}{\Longrightarrow} (Q', \varphi') \wedge \varphi \sim_s \varphi'$.

$P \approx_t Q$ iff $P \sqsubseteq_t Q \wedge Q \sqsubseteq_t P$.

Trace equivalences

Fine grained trace equivalence

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

$P \approx_{ft} Q$ iff $P \sqsubseteq_{ft} Q \wedge Q \sqsubseteq_{ft} P$.

Trace equivalence

$P \sqsubseteq_t Q$ if $(P, \emptyset) \stackrel{\ell_1,\ldots,\ell_n}{\Longrightarrow} (P', \varphi)$ implies

$\exists Q', \varphi'.\ (Q, \emptyset) \stackrel{\ell_1,\ldots,\ell_n}{\Longrightarrow} (Q', \varphi') \wedge \varphi \sim_s \varphi'$.

$P \approx_t Q$ iff $P \sqsubseteq_t Q \wedge Q \sqsubseteq_t P$.

Trace equivalences

Fine grained trace equivalence

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

$P \approx_{ft} Q$ iff $P \sqsubseteq_{ft} Q \wedge Q \sqsubseteq_{ft} P$.

Trace equivalence

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

$P \approx_t Q$ iff $P \sqsubseteq_t Q \wedge Q \sqsubseteq_t P$.

Example:

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

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

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

Trace equivalences

Fine grained trace equivalence

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

$P \approx_{ft} Q$ iff $P \sqsubseteq_{ft} Q \wedge Q \sqsubseteq_{ft} P$.

Trace equivalence

$P \sqsubseteq_t Q$ if $(P, \emptyset) \stackrel{\ell_1,\ldots,\ell_n}{\Longrightarrow} (P', \varphi)$ implies

$\exists Q', \varphi'.\ (Q, \emptyset) \stackrel{\ell_1,\ldots,\ell_n}{\Longrightarrow} (Q', \varphi') \wedge \varphi \sim_s \varphi'$.

$P \approx_t Q$ iff $P \sqsubseteq_t Q \wedge Q \sqsubseteq_t P$.

Coarse trace equivalence

$P \sqsubseteq_{ct} Q$ if $(P, \emptyset) \stackrel{\ell_1,\ldots,\ell_n}{\Longrightarrow} (P', \varphi)$ implies

$\exists Q', \varphi'.\ (Q, \emptyset) \stackrel{\ell_1,\ldots,\ell_n}{\Longrightarrow} (Q', \varphi') \wedge \varphi \sqsubseteq_s \varphi'$.

$P \approx_{ct} Q$ iff $P \sqsubseteq_{ct} Q \wedge Q \sqsubseteq_{ct} P$.

Trace equivalences

Fine grained trace equivalence

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

$P \approx_{ft} Q$ iff $P \sqsubseteq_{ft} Q \wedge Q \sqsubseteq_{ft} P$.

Trace equivalence

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

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

$P \approx_t Q$ iff $P \sqsubseteq_t Q \wedge Q \sqsubseteq_t P$.

Coarse trace equivalence

$P \sqsubseteq_{ct} Q$ if $(P, \emptyset) \xRightarrow{\ell_1,\ldots,\ell_n} (P', \varphi)$ implies

$\exists Q', \varphi'.\ (Q, \emptyset) \xRightarrow{\ell_1,\ldots,\ell_n} (Q', \varphi') \wedge \varphi \sqsubseteq_s \varphi'$.

$P \approx_{ct} Q$ iff $P \sqsubseteq_{ct} Q \wedge Q \sqsubseteq_{ct} P$.

Example:

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

$P = \{\mathsf{send}(c, a).\mathsf{send}(c, a)\}$

$Q = \{\mathsf{send}(c, a).\mathsf{send}(c, a),$
$\mathsf{send}(c, a).\mathsf{send}(c, b)\}$.

Trace equivalences

Fine grained trace equivalence

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

$P \approx_{ft} Q$ iff $P \sqsubseteq_{ft} Q \wedge Q \sqsubseteq_{ft} P$.

Trace equivalence

$P \sqsubseteq_t Q$ if $(P, \emptyset) \stackrel{\ell_1,\ldots,\ell_n}{\Longrightarrow} (P', \varphi)$ implies

$\exists Q', \varphi'.\ (Q, \emptyset) \stackrel{\ell_1,\ldots,\ell_n}{\Longrightarrow} (Q', \varphi') \wedge \varphi \sim_s \varphi'$.

$P \approx_t Q$ iff $P \sqsubseteq_t Q \wedge Q \sqsubseteq_t P$.

Coarse trace equivalence

$P \sqsubseteq_{ct} Q$ if $(P, \emptyset) \stackrel{\ell_1,\ldots,\ell_n}{\Longrightarrow} (P', \varphi)$ implies

$\exists Q', \varphi'.\ (Q, \emptyset) \stackrel{\ell_1,\ldots,\ell_n}{\Longrightarrow} (Q', \varphi') \wedge \varphi \sqsubseteq_s \varphi'$.

$P \approx_{ct} Q$ iff $P \sqsubseteq_{ct} Q \wedge Q \sqsubseteq_{ct} P$.

Definition:

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

Trace equivalences

Fine grained trace equivalence

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

$P \approx_{ft} Q$ iff $P \sqsubseteq_{ft} Q \wedge Q \sqsubseteq_{ft} P$.

Trace equivalence

$P \sqsubseteq_t Q$ if $(P, \emptyset) \stackrel{\ell_1,\ldots,\ell_n}{\Longrightarrow} (P', \varphi)$ implies

$\exists Q', \varphi'.\ (Q, \emptyset) \stackrel{\ell_1,\ldots,\ell_n}{\Longrightarrow} (Q', \varphi') \wedge \varphi \sim_s \varphi'$.

$P \approx_t Q$ iff $P \sqsubseteq_t Q \wedge Q \sqsubseteq_t P$.

det. proc.

Coarse trace equivalence

$P \sqsubseteq_{ct} Q$ if $(P, \emptyset) \stackrel{\ell_1,\ldots,\ell_n}{\Longrightarrow} (P', \varphi)$ implies

$\exists Q', \varphi'.\ (Q, \emptyset) \stackrel{\ell_1,\ldots,\ell_n}{\Longrightarrow} (Q', \varphi') \wedge \varphi \sqsubseteq_s \varphi'$.

$P \approx_{ct} Q$ iff $P \sqsubseteq_{ct} Q \wedge Q \sqsubseteq_{ct} P$.

Remark:

A trace is a determinate process

Definition:

P is *determinate* if whenever $(P, \emptyset) \stackrel{\ell_1,\ldots,\ell_n}{\Longrightarrow} (T, \varphi)$ and $(P, \emptyset) \stackrel{\ell_1,\ldots,\ell_n}{\Longrightarrow} (T', \varphi')$ 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

1. Horn clause modelling

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

Compute a initial set for trace T: $\mathsf{seed}(T)$

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

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

1. Horn clause modelling

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

Compute a initial set for trace T: $\mathsf{seed}(T)$

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

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

Add history for accuracy (avoid false attacks)

1. Horn clause modelling

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

Compute a initial set for trace T: $\mathsf{seed}(T)$

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

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

Add history for accuracy (avoid false attacks)

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

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

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 $u =_{\mathsf{R}} v$.

Use finite variant property to get rid of equational reasoning:

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

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

We can compute finite sets of variants and $\mathsf{mgu}_{\mathsf{E}}$ 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 $(T,\emptyset) \xrightarrow{L_1} (T_1,\varphi_1) \xrightarrow{L_2} \ldots \xrightarrow{L_n} (T_n,\varphi_n)$
  such that $\ell_i =_{\mathsf{R}} L_i\varphi_{i-1}$ for all $1 \leq i \leq n$
  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 $(T, \emptyset) \xrightarrow{L_1,\ldots,L_m} (S, \varphi)$ then*
  1. $\mathsf{r}_{L_1\varphi\downarrow,\ldots,L_m\varphi\downarrow} \in \mathcal{H}(\mathsf{seed}(T))$, *and*
  2. *if $\varphi \vdash^R t$ then $\mathsf{k}_{L_1\varphi\downarrow,\ldots,L_m\varphi\downarrow}(R, t\downarrow) \in \mathcal{H}(\mathsf{seed}(T))$.*

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

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

$$\text{Equation}\ \frac{\begin{array}{c} f, g \in K_{\text{solved}}, \qquad f = \Big(\mathsf{k}_u(R,t) \Leftarrow B_1, \ldots, B_n\Big) \\ g = \Big(\mathsf{k}_{u'v'}(R',t') \Leftarrow B_{n+1}, \ldots, B_m\Big) \\ \sigma = \text{mgu}(\mathsf{k}_u(\_,t), \mathsf{k}_{u'}(\_,t')) \end{array}}{K = K \oplus h \text{ where } h = \Big((\mathsf{i}_{u'v'}(R,R') \Leftarrow B_1, \ldots, B_m)\sigma\Big)}$$

$$\text{Test}\ \frac{\begin{array}{c} f, g \in K_{\text{solved}}, \qquad f = \Big(\mathsf{i}_u(R,R') \Leftarrow B_1, \ldots, B_n\Big) \\ g = \Big(\mathsf{r}_{u'v'} \Leftarrow B_{n+1}, \ldots, B_m\Big) \qquad \sigma = \text{mgu}(u,u') \end{array}}{K = K \oplus h \text{ where } h = \Big((\mathsf{ri}_{u'v'}(R,R') \Leftarrow B_1, \ldots, B_m)\sigma\Big)}$$

2. Saturation rules: soundness, completeness, termination

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

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. $\mathsf{r}_{L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow} \in \mathcal{H}_e(K_{\mathsf{solved}})$
  2. if $\varphi \vdash^R t$ then $\mathsf{k}_{L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow}(R, t\downarrow) \in \mathcal{H}_e(K_{\mathsf{solved}})$
  3. if $\varphi \vdash^R t$ and $\varphi \vdash^{R'} t$, then $\mathsf{i}_{L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow}(R, R') \in \mathcal{H}_e(K_{\mathsf{solved}})$

  where $\mathcal{H}_e(K)$ the smallest set of ground terms such that
  - $\mathcal{H}(K) \subseteq \mathcal{H}_e(K)$,
  - $\mathcal{H}_e(K)$ is closed under congruence rules for each $\mathsf{i}_w(R, R') \in \mathcal{H}_e(K)$,
  - 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. $\mathsf{r}_{L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow} \in \mathcal{H}_e(K_{\mathsf{solved}})$
  2. if $\varphi \vdash^R t$ then $\mathsf{k}_{L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow}(R, t\downarrow) \in \mathcal{H}_e(K_{\mathsf{solved}})$
  3. if $\varphi \vdash^R t$ and $\varphi \vdash^{R'} t$, then $\mathsf{i}_{L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow}(R, R') \in \mathcal{H}_e(K_{\mathsf{solved}})$

  where $\mathcal{H}_e(K)$ the smallest set of ground terms such that
  - $\mathcal{H}(K) \subseteq \mathcal{H}_e(K)$,
  - $\mathcal{H}_e(K)$ is closed under congruence rules for each $\mathsf{i}_w(R, R') \in \mathcal{H}_e(K)$,
  - 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 $K = \mathsf{sat}(\mathsf{seed}(T))_{\mathsf{solved}}$
2. check reachability:
   for each $\mathsf{r}_{L_1,\ldots,L_n} \Leftarrow \mathsf{k}_{h_1}(X_1, x_1), \ldots \mathsf{k}_{h_k}(X_k, x_k) \in K$
   such that for all $1 \leq i \leq n$ $\mathsf{k}_{h_i}(X_i, x_i) \in \mathcal{H}(K)$
   check that $Q, \emptyset \xrightarrow{L_1,\ldots,L_n} Q', \varphi$
3. check static inclusion:
   for each $\mathsf{ri}_{L_1,\ldots,L_n}(R_1, R_2) \Leftarrow \mathsf{k}_{h_1}(X_1, x_1), \ldots \mathsf{k}_{h_k}(X_k, x_k) \in K$
   such that for all $1 \leq i \leq n$ $\mathsf{k}_{h_i}(X_i, x_i) \in \mathcal{H}(K)$
   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)

`http://www.lsv.ens-cachan.fr/~ciobaca/akiss/`

$\sim$2000 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 automated proof
  (non-determinate processes $\rightsquigarrow$ proof of $\approx_{ft}$)

Conclusion

- An accurate horn clause based modelling of protocols for a bounded number of sessions
- Procedure for checking (some) equivalence properties
  (easily checks deduction-based secrecy via knows predicate)
- Prototype implementation:
  (anonymity in FOO and Okamoto electronic voting protocol, strong secrecy of NSPK, resistance to guessing attacks of EKE, ...)

A lot of work to be done

- Termination? (but does it matter?)
- More examples and evaluation of the prototype
- "Real" trace equivalence (can we add disequalities?)
- Negative tests (else branches)
- Performance: optimize, parallelize, check bisimulation
  examples may take a few mins on a laptop (e.g. $\sim$2 mins for FOO)