Symbolic Verification of Cryptographic Protocols

Protocol Equivalences

David Baelde

LSV, ENS Paris-Saclay & Prosecco, Inria Paris

2017

Static equivalence

The first equivalence does not involve process executions,
but only sequences of messages.

When are two sequences of message distinguishable?

Examples

- $\langle u, v, v \rangle \sim \langle v, u, v \rangle$?

Static equivalence

The first equivalence does not involve process executions,
but only sequences of messages.

When are two sequences of message distinguishable?

Examples

- $\langle u, v, v \rangle \sim \langle v, u, v \rangle$?
- $\langle n \rangle \sim \langle n' \rangle$? $\langle \langle n, m \rangle \rangle \sim \langle \langle n', n' \rangle \rangle$?

Static equivalence

The first equivalence does not involve process executions,
but only sequences of messages.

When are two sequences of message distinguishable?

Examples

- $\langle u, v, v\rangle \sim \langle v, u, v\rangle$?
- $\langle n\rangle \sim \langle n'\rangle$? $\langle\langle n, m\rangle\rangle \sim \langle\langle n', n'\rangle\rangle$?
- $\langle\langle u, v\rangle\rangle \sim \langle n'\rangle$? $\langle \mathsf{senc}(u, k)\rangle \sim \langle n'\rangle$?

Static equivalence

The first equivalence does not involve process executions,
but only sequences of messages.

When are two sequences of message distinguishable?

Examples

- $\langle u, v, v \rangle \sim \langle v, u, v \rangle$?
- $\langle n \rangle \sim \langle n' \rangle$? $\langle \langle n, m \rangle \rangle \sim \langle \langle n', n' \rangle \rangle$?
- $\langle \langle u, v \rangle \rangle \sim \langle n' \rangle$? $\langle \mathsf{senc}(u, k) \rangle \sim \langle n' \rangle$?
- $\langle \mathsf{senc}(u, k) \rangle \sim \langle \mathsf{senc}(v, k) \rangle$? $\langle \mathsf{senc}(u, k) \rangle \sim \langle \mathsf{senc}(u, k') \rangle$?

Static equivalence

The first equivalence does not involve process executions,
but only sequences of messages.

When are two sequences of message distinguishable?

Examples

- $\langle u, v, v \rangle \sim \langle v, u, v \rangle$?
- $\langle n \rangle \sim \langle n' \rangle$? $\langle \langle n, m \rangle \rangle \sim \langle \langle n', n' \rangle \rangle$?
- $\langle \langle u, v \rangle \rangle \sim \langle n' \rangle$? $\langle \mathsf{senc}(u, k) \rangle \sim \langle n' \rangle$?
- $\langle \mathsf{senc}(u, k) \rangle \sim \langle \mathsf{senc}(v, k) \rangle$? $\langle \mathsf{senc}(u, k) \rangle \sim \langle \mathsf{senc}(u, k') \rangle$?
- $\langle \mathsf{aenc}(u, pk), u, pk \rangle \sim \langle \mathsf{aenc}(v, pk), u, pk \rangle$?

Static equivalence

Definition

A frame $\Phi : \mathcal{W} \to \mathcal{M}$ is a substitution associating messages $u, v \in \mathcal{M} = \mathcal{T}(\Sigma_c, \mathcal{N})$ to handles (special variables $w \in \mathcal{W}$).

Definition

Two frames Φ_1 and Φ_2 are statically equivalent when

- they have the same domain: $\mathrm{dom}(\Phi_1) = \mathrm{dom}(\Phi_2)$;
- for all $M, N \in \mathcal{T}(\Sigma, \mathcal{W})$, $M\Phi_1 =_{\mathsf{E}} N\Phi_1$ iff $M\Phi_2 =_{\mathsf{E}} N\Phi_2$.

Proposition

Static equivalence is an equivalence. It is stable by bijective renaming. It does not compose: $\Phi_1 \sim \Phi_1'$ *and* $\Phi_2 \sim \Phi_2' \not\Rightarrow \Phi_1 \uplus \Phi_2 \sim \Phi_1' \uplus \Phi_2'$.

Static equivalence: examples

Suppose we are in Σ_{std} with Estd.

Examples (bis)

- $\{w_1 \mapsto u, w_2 \mapsto v, w_3 \mapsto v\} \sim \{w_1 \mapsto v, w_2 \mapsto u, w_3 \mapsto v\}$?
- $\{w \mapsto n\} \sim \{w \mapsto n'\}$? $\{w \mapsto \langle n, m\rangle\} \sim \{w \mapsto \langle n', n'\rangle\}$?
- $\{w \mapsto \langle u, v\rangle\} \sim \{w \mapsto n'\}$? $\{w \mapsto \mathsf{senc}(u, k)\} \sim \{w \mapsto n'\}$?
- $\{w \mapsto \mathsf{senc}(u, k)\} \sim \{w \mapsto \mathsf{senc}(v, k)\}$?
  $\{w \mapsto \mathsf{senc}(u, k)\} \sim \{w \mapsto \mathsf{senc}(u, k')\}$?
- $\{w \mapsto \mathsf{aenc}(u, pk), w' \mapsto u, w'' \mapsto pk\} \sim$
  $\{w \mapsto \mathsf{aenc}(v, pk), w' \mapsto u, w'' \mapsto pk\}$?

Application: guessing attacks

We usually assume that secrets cannot be guessed: no brute force attacks.

That is not reasonable for low/fixed entropy secrets, such as PIN, passwords, one-time verification code, etc.

Offline guessing attacks

A protocol is resistant against offline guessing attacks on some name d when any reachable frame Φ is such that

$$\Phi \cup \{w \mapsto d\} \sim \Phi \cup \{w \mapsto d'\} \text{ for } w, d' \text{ fresh.}$$

This notion is meaningful even with a passive adversary.

Application: EKE

Assume public-key encryption but no PKI (public keys $\neq$ identities).
A and B only share a weak password p, want to authenticate.

$$
\begin{array}{lll}
1. & A \rightarrow B: & \mathsf{senc}(\mathsf{pub}(k), p) \\
2. & B \rightarrow A: & \mathsf{senc}(\mathsf{aenc}(r, \mathsf{pub}(k)), p) \\
3. & A \rightarrow B: & \mathsf{senc}(n_a, r) \\
4. & B \rightarrow A: & \mathsf{senc}(\langle n_a, n_b \rangle, r) \\
5. & A \rightarrow B: & \mathsf{senc}(n_b, r)
\end{array}
$$

Application: EKE

Assume public-key encryption but no PKI (public keys $\neq$ identities).
A and B only share a weak password p, want to authenticate.

$$\begin{array}{lll} 1. & A \rightarrow B: & \mathsf{senc}(\mathsf{pub}(k), p) \\ 2. & B \rightarrow A: & \mathsf{senc}(\mathsf{aenc}(r, \mathsf{pub}(k)), p) \\ 3. & A \rightarrow B: & \mathsf{senc}(n_a, r) \\ 4. & B \rightarrow A: & \mathsf{senc}(\langle n_a, n_b \rangle, r) \\ 5. & A \rightarrow B: & \mathsf{senc}(n_b, r) \end{array}$$

Let $\Phi = \{w_1 \mapsto \mathsf{senc}(\mathsf{pub}(k), p), \ \ldots, \ w_5 \mapsto \mathsf{senc}(n_b, r)\}$.
Can p be guessed offline, that is

$$\Phi \cup \{w \mapsto p\} \ \sim \ \Phi \cup \{w \mapsto p'\} \ ?$$

Application: EKE

Assume public-key encryption but no PKI (public keys $\neq$ identities).
A and B only share a weak password p, want to authenticate.

$$\begin{array}{lll} 1. & A \to B: & \mathsf{senc}(\mathsf{pub}(k), p) \\ 2. & B \to A: & \mathsf{senc}(\mathsf{aenc}(r, \mathsf{pub}(k)), p) \\ 3. & A \to B: & \mathsf{senc}(n_a, r) \\ 4. & B \to A: & \mathsf{senc}(\langle n_a, n_b \rangle, r) \\ 5. & A \to B: & \mathsf{senc}(n_b, r) \end{array}$$

Let $\Phi = \{w_1 \mapsto \mathsf{senc}(\mathsf{pub}(k), p), \ \ldots, \ w_5 \mapsto \mathsf{senc}(n_b, r)\}$.
Can p be guessed offline, that is

$$\Phi \cup \{w \mapsto p\} \ \sim \ \Phi \cup \{w \mapsto p'\} \ ?$$

Only if $\mathsf{senc}(\mathsf{sdec}(x, y), y) = x$.

May testing

The reduction semantics (cf. previous lectures) provide a first natural definition of when two processes can be distinguished.

Definition

A test is a process with no free name and in which a special channel $\mathbb{T}$ may occur. A process P may pass a test T, written $P \models T$ if

$$P \mid T \rightsquigarrow^* \mathsf{out}(\mathbb{T}, u) \mid Q \quad \text{for some } u \text{ and } Q.$$

Let $\mathrm{T}(P) := \{\ T \mid P \models T\ \}$.
Processes P and Q are in may-testing equivalence when $\mathrm{T}(P) = \mathrm{T}(Q)$.

May testing

The reduction semantics (cf. previous lectures) provide a first natural definition of when two processes can be distinguished.

Definition

A test is a process with no free name and in which a special channel $\mathbb{T}$ may occur. A process P may pass a test T, written $P \models T$ if

$$P \mid T \rightsquigarrow^* \mathsf{out}(\mathbb{T}, u) \mid Q \quad \text{for some } u \text{ and } Q.$$

Let $\mathsf{T}(P) := \{\ T \mid P \models T\ \}$.
Processes P and Q are in may-testing equivalence when $\mathsf{T}(P) = \mathsf{T}(Q)$.

Quite natural, but may not model all desired aspects, e.g. probabilities, must testing, asynchronicity.

May testing

The reduction semantics (cf. previous lectures) provide a first natural definition of when two processes can be distinguished.

Definition

A test is a process with no free name and in which a special channel $\mathbb{T}$ may occur. A process P may pass a test T, written $P \models T$ if

$$P \mid T \rightsquigarrow^* \mathsf{out}(\mathbb{T}, u) \mid Q \quad \text{for some } u \text{ and } Q.$$

Let $\mathrm{T}(P) := \{\ T \mid P \models T\ \}$.
Processes P and Q are in may-testing equivalence when $\mathrm{T}(P) = \mathrm{T}(Q)$.

Quite natural, but may not model all desired aspects, e.g. probabilities, must testing, asynchronicity.
As such, may testing equivalence is hard to verify !

Labelled transition system with handles

A configuration is a pair (P, Φ) where P is a ground process and $\Phi : \mathcal{W} \to \mathcal{M}$ is a frame.

$$(\mathsf{out}(c, u).P \mid Q, \Phi) \xrightarrow{\mathsf{out}(c,w)} (P \mid Q, \Phi \cup \{w \mapsto u\})$$

where w is fresh, $u =_{\mathsf{E}} v \in \mathcal{M}$

$$(\mathsf{in}(c, x).P \mid Q, \Phi) \xrightarrow{\mathsf{in}(c,M)} (P[x := u] \mid Q, \Phi)$$

where $u \in \mathcal{M}$, $u =_{\mathsf{E}} M\Phi$ for some $M \in \mathcal{T}(\Sigma, \mathcal{W})$

$$(\mathsf{if}\ u = v\ \mathsf{then}\ P\ \mathsf{else}\ Q \mid R, \Phi) \xrightarrow{\tau} (P \mid R, \Phi) \qquad \text{when } u =_{\mathsf{E}} v$$

$$(\mathsf{if}\ u = v\ \mathsf{then}\ P\ \mathsf{else}\ Q \mid R, \Phi) \xrightarrow{\tau} (Q \mid R, \Phi) \qquad \text{when } u \neq_{\mathsf{E}} v$$

$$((\mathsf{new}\ x.P) \mid Q, \Phi) \xrightarrow{\tau} (P[x := n] \mid Q, \Phi) \qquad \text{for some fresh } n$$

$$(!P \mid Q, \Phi) \xrightarrow{\tau} (P \mid !P \mid Q, \Phi)$$

Trace equivalence

Weak labelled transitions

We write $A \stackrel{\mathsf{tr}}{\Rightarrow} B$ when:

- tr only contains input and output actions (no τ);
- there exists tr′ obtained from tr by adding τs such that $A \xrightarrow{\mathsf{tr}'} B$.

Definition

Given a configuration $A = (P, \Phi)$, define

$$\mathsf{Tr}(A) := \{\ (\mathsf{tr}, \Phi') \mid A \stackrel{\mathsf{tr}}{\Rightarrow} (\_, \Phi')\ \}.$$

We say that A and B are trace equivalent, noted $A \approx B$, iff

$$\text{for all } (\mathsf{tr}, \Phi') \in \mathsf{Tr}(A) \text{ there exists } (\mathsf{tr}, \Psi') \in \mathsf{Tr}(B).\ \ \Phi' \sim \Psi'$$

and conversely.

Alternative definition

Proposition

Close $\mathrm{Tr}(\cdot)$ *under static equivalence:*

$$\mathrm{Tr}'(P,\Phi) := \{\ \mathrm{tr}, \Phi' \mid (P,\Phi) \stackrel{\mathrm{tr}}{\Rightarrow} (P',\Phi''),\ \Phi'' \sim \Phi'\ \}$$

Then we have $A \approx B$ *iff* $\mathrm{Tr}'(A) = \mathrm{Tr}'(B)$*.*

Remarks

$A \approx B$ imposes $\Phi(A) \sim \Phi(B)$ and thus $\mathrm{dom}(\Phi(A)) = \mathrm{dom}(\Phi(B))$,

Alternative definition

Proposition

Close $\mathrm{Tr}(\cdot)$ *under static equivalence:*

$$\mathrm{Tr}'(P,\Phi) := \{\ \mathrm{tr}, \Phi' \mid (P,\Phi) \stackrel{\mathrm{tr}}{\Rightarrow} (P',\Phi''),\ \Phi'' \sim \Phi'\ \}$$

Then we have $A \approx B$ *iff* $\mathrm{Tr}'(A) = \mathrm{Tr}'(B)$.

Remarks

$A \approx B$ imposes $\Phi(A) \sim \Phi(B)$ and thus $\mathrm{dom}(\Phi(A)) = \mathrm{dom}(\Phi(B))$, but not $\Phi(A) = \Phi(B)$.

Alternative definition

Proposition

Close $\mathrm{Tr}(\cdot)$ *under static equivalence:*

$$\mathrm{Tr}'(P,\Phi) := \{\ \mathrm{tr}, \Phi' \mid (P,\Phi) \stackrel{\mathrm{tr}}{\Rightarrow} (P',\Phi''),\ \Phi'' \sim \Phi'\ \}$$

Then we have $A \approx B$ *iff* $\mathrm{Tr}'(A) = \mathrm{Tr}'(B)$*.*

Remarks

$A \approx B$ imposes $\Phi(A) \sim \Phi(B)$ and thus $\mathrm{dom}(\Phi(A)) = \mathrm{dom}(\Phi(B))$, but not $\Phi(A) = \Phi(B)$.
In general we do not have that $\Phi \sim \Psi$ implies $(P,\Phi) \approx (P,\Psi)$.

Examples

1. $\mathsf{in}(c,x).\mathsf{out}(c,\mathsf{ok}) \approx^? \mathsf{in}(c,x) \mid \mathsf{out}(c,\mathsf{ok})$
2. $\mathsf{in}(c,x).\mathsf{out}(c,\mathsf{ok}) \approx^? \mathsf{in}(c,x).\mathsf{out}(c,x)$
3. $\mathsf{new}\ n,\ m.\ \mathsf{out}(c,n) \mid \mathsf{out}(c,m) \approx^? \mathsf{new}\ n,\ m.\ \mathsf{out}(c,n).\mathsf{out}(c,m)$
4. $\mathsf{new}\ n,m.\ \mathsf{out}(c,n) \mid \mathsf{out}(c,\mathsf{hash}(m)) \approx^?$
   $\mathsf{new}\ n.\ \mathsf{out}(c,n).\mathsf{out}(c,\mathsf{hash}(n))$
5. $\mathsf{out}(c,u_1).\ldots.\mathsf{out}(c,u_n).\mathsf{in}(c,x).\mathsf{if}\ x = v\ \mathsf{then}\ \mathsf{out}(c,\mathsf{ok}) \approx^?$
   $\mathsf{out}(c,u_1).\ldots.\mathsf{out}(c,u_n).\mathsf{in}(c,x).0$

Trace equivalence $\subseteq$ may-testing

Proposition

If $(P, \emptyset) \approx (Q, \emptyset)$ *then they are in may-testing equivalence.*

Proof sketch.

Assume $P \models T$. There is a sequence of $(P_i, \alpha_i, \Phi_i, T_i)_{1 \leq i \leq n}$ such that

- $(P, T) = (P_0, T_0)$, $\Phi_0 = \emptyset$,
- T_i contains terms in $\mathcal{T}(\Sigma, \mathcal{X} \cup \mathrm{dom}(\Phi_i))$,
- $T_n \Phi_n \rightsquigarrow^* \mathrm{out}(\mathbb{T}, \_) \mid \_$,
- $P_i \mid T_i \Phi_i \rightsquigarrow^* P_{i+1} \mid T_{i+1} \Phi_{i+1}$ with exactly one communication between P_i and $T_i \Phi_i$, and no communication within P_i (wlog);
- $(P_i, \Phi_i) \xrightarrow{\alpha_i} (P_{i+1}, \Phi_{i+1})$.

Then $(\alpha_1 \cdots \alpha_n, \Phi_n) \in \mathrm{Tr}(P, \emptyset)$, thus $(\alpha_1 \cdots \alpha_n, \Phi_n) \in \mathrm{Tr}'(Q, \emptyset)$.
We conclude by showing that $Q \mid T = Q_0 \mid T_0 \Psi_0 \rightsquigarrow^* \ldots Q_n \mid T_n \Psi_n$. □

May testing $\not\subseteq$ trace equivalence

Proposition

If P and Q are may-testing equivalent then $P \approx Q$, ...

May testing $\not\subseteq$ trace equivalence

Proposition

If P and Q are may-testing equivalent then $P \approx Q$, provided the processes are image-finite:

$$\textit{for any } \mathsf{tr},\ \{\ \Phi \mid (\mathsf{tr}, \Phi) \in \mathsf{Tr}'(P, \emptyset)\ \}\ \textit{is finite up to } \sim$$

and similarly for Q.

May testing $\not\subseteq$ trace equivalence

Proposition

If P and Q are may-testing equivalent then $P \approx Q$, provided the processes are image-finite:

$$\textit{for any } \mathsf{tr},\ \{\ \Phi \mid (\mathsf{tr}, \Phi) \in \mathsf{Tr}'(P, \emptyset)\ \}\ \textit{is finite up to } \sim$$

and similarly for Q.

Example

$$P := \mathsf{new}\ c.\ (\mathsf{out}(c, \mathsf{ok}) \mid !\ \mathsf{in}(c, x).\mathsf{out}(c, h(x)) \mid \mathsf{in}(c, x).\mathsf{out}(a, x))$$

$$Q := P \mid \mathsf{new}\ n.\ \mathsf{out}(a, n)$$

We have $P \not\approx Q$ but P and Q are in may-testing equivalence.

May testing $\not\subseteq$ trace equivalence

Proposition

If P and Q are may-testing equivalent then $P \approx Q$, provided the processes are image-finite:

$$\textit{for any } \mathsf{tr},\ \{\ \Phi \mid (\mathsf{tr}, \Phi) \in \mathsf{Tr}'(P, \emptyset)\ \}\ \textit{is finite up to}\ \sim$$

and similarly for Q.

Example

$$P := \mathsf{new}\ c.\ (\mathsf{out}(c, \mathsf{ok}) \mid !\ \mathsf{in}(c, x).\mathsf{out}(c, h(x)) \mid \mathsf{in}(c, x).\mathsf{out}(a, x))$$

$$Q := P \mid \mathsf{new}\ n.\ \mathsf{out}(a, n)$$

We have $P \not\approx Q$ but P and Q are in may-testing equivalence.

This is “only” pathological !

Application: strong secrecy

Definition

A protocol P ensures the strong secrecy of some variables $\vec{x}$ if, for all (relevant) values $\vec{u}$, $\vec{v}$, $P[\vec{x} := \vec{u}] \approx P[\vec{x} := \vec{v}]$.

Weak secrecy: some value cannot be (fully) derived by the attacker.
Strong secrecy: the attacker has no information at all about the value.

Application: strong secrecy

Definition

A protocol P ensures the strong secrecy of some variables $\vec{x}$ if, for all (relevant) values $\vec{u}$, $\vec{v}$, $P[\vec{x} := \vec{u}] \approx P[\vec{x} := \vec{v}]$.

Weak secrecy: some value cannot be (fully) derived by the attacker.
Strong secrecy: the attacker has no information at all about the value.

Blanchet's key exchange protocol:

$$\begin{array}{lll} 1. & A \to B: & \mathsf{aenc}(\mathsf{sign}(\langle pk_A, pk_B, k\rangle, sk_A), pk_B) \\ 2. & B \to A: & \mathsf{senc}(x, k) \\ 3. & A \to B: & \mathsf{senc}(y, k) \end{array}$$

Scenario: A and B honest. Is x strongly secret? Are x, y strongly secret?

Application: private authentication

Agents A and B want to authenticate, without revealing their identities.

$I(sk_a, pk_b)$	$R(sk_b, pk_a)$
$\text{new } n_a.$	$\text{new } n_b.$
$\text{let } pk_a = \text{pub}(sk_a) \text{ in}$	$\text{let } pk_b = \text{pub}(sk_b) \text{ in}$
$\text{out}(c, \text{aenc}(\langle n_a, pk_a\rangle, pk_b)).$	$\text{in}(c, x).\text{let } y = \text{adec}(x, sk_b) \text{ in}$
$\ldots$	$\text{if } \text{proj}_2(y) = pk_a \text{ then}$
	$\text{out}(c, \text{aenc}(\langle \text{proj}_1(y), n_b, pk_b\rangle, pk_a))$

Anonymity

$$\text{new } sk_a, sk_b, sk_c.\ \text{out}(c, \langle \text{pub}(sk_a), \text{pub}(sk_b), \text{pub}(sk_c)\rangle).R(sk_b, \text{pub}(sk_a))$$
$$\approx^?$$
$$\text{new } sk_a, sk_b, sk_c.\ \text{out}(c, \langle \text{pub}(sk_a), \text{pub}(sk_b), \text{pub}(sk_c)\rangle).R(sk_b, \text{pub}(sk_c))$$

Application: private authentication

Agents A and B want to authenticate, without revealing their identities.

$I(sk_a, pk_b)$	$R(sk_b, pk_a)$
$\text{new } n_a.$	$\text{new } n_b.$
$\text{let } pk_a = \text{pub}(sk_a) \text{ in}$	$\text{let } pk_b = \text{pub}(sk_b) \text{ in}$
$\text{out}(c, \text{aenc}(\langle n_a, pk_a \rangle, pk_b)).$	$\text{in}(c, x).\text{let } y = \text{adec}(x, sk_b) \text{ in}$
$\ldots$	$\text{if } \text{proj}_2(y) = pk_a \text{ then}$
	$\text{out}(c, \text{aenc}(\langle \text{proj}_1(y), n_b, pk_b \rangle, pk_a))$
	$\text{else out}(c, \text{aenc}(n_b, pk_b)) \ \leftarrow$ decoy!

Anonymity

$$\text{new } sk_a, sk_b, sk_c.\ \text{out}(c, \langle \text{pub}(sk_a), \text{pub}(sk_b), \text{pub}(sk_c) \rangle).R(sk_b, \text{pub}(sk_a))$$
$$\approx^?$$
$$\text{new } sk_a, sk_b, sk_c.\ \text{out}(c, \langle \text{pub}(sk_a), \text{pub}(sk_b), \text{pub}(sk_c) \rangle).R(sk_b, \text{pub}(sk_c))$$

Application: unlinkability

The BAC e-passport protocol is used between a tag T and a reader R.
After k_E and k_M are derived from optical scan (shared secrets), a key is established as follows:

$$\begin{array}{lll} 1. & T \to R: & n_T \\ 2. & R \to T: & \mathsf{senc}(\langle n_R, n_T, k_R \rangle, k_E), \mathsf{mac}(\mathsf{senc}(\langle n_R, n_T, k_R \rangle, k_E), k_M) \\ 3. & T \to R: & \mathsf{senc}(\langle n_T, n_R, k_T \rangle, k_E), \mathsf{mac}(\mathsf{senc}(\langle n_T, n_R, k_T \rangle, k_E), k_M) \end{array}$$

Application: unlinkability

The BAC e-passport protocol is used between a tag T and a reader R.
After k_E and k_M are derived from optical scan (shared secrets),
a key is established as follows:

$$
\begin{array}{lll}
1. & T \to R: & n_T \\
2. & R \to T: & \mathsf{senc}(\langle n_R, n_T, k_R \rangle, k_E), \mathsf{mac}(\mathsf{senc}(\langle n_R, n_T, k_R \rangle, k_E), k_M) \\
3. & T \to R: & \mathsf{senc}(\langle n_T, n_R, k_T \rangle, k_E), \mathsf{mac}(\mathsf{senc}(\langle n_T, n_R, k_T \rangle, k_E), k_M)
\end{array}
$$

French implementation:

$$
\begin{array}{rl}
T(k_E, k_M) := & \mathsf{new}\ n_T, k_T.\ \mathsf{out}(c, n_T).\mathsf{in}(c, x). \\
& \mathsf{if}\ \mathsf{mac}(\mathsf{proj}_1(x), k_M) = \mathsf{proj}_2(x)\ \mathsf{then} \\
& \quad \mathsf{if}\ n_T = \mathsf{proj}_1(\mathsf{sdec}(\mathsf{proj}_1(x), k_E))\ \mathsf{then}\ \ldots\ \mathsf{else} \\
& \quad \mathsf{out}(c, \mathrm{ERR\_nonce}) \\
& \mathsf{elseout}(c, \mathrm{ERR\_mac})
\end{array}
$$

Application: unlinkability

The BAC e-passport protocol is used between a tag T and a reader R.
After k_E and k_M are derived from optical scan (shared secrets),
a key is established as follows:

$$
\begin{array}{lll}
1. & T \to R: & n_T \\
2. & R \to T: & \mathsf{senc}(\langle n_R, n_T, k_R\rangle, k_E), \mathsf{mac}(\mathsf{senc}(\langle n_R, n_T, k_R\rangle, k_E), k_M) \\
3. & T \to R: & \mathsf{senc}(\langle n_T, n_R, k_T\rangle, k_E), \mathsf{mac}(\mathsf{senc}(\langle n_T, n_R, k_T\rangle, k_E), k_M)
\end{array}
$$

French implementation:

$$
\begin{array}{rl}
T(k_E, k_M) := & \mathsf{new}\ n_T, k_T.\ \mathsf{out}(c, n_T).\mathsf{in}(c, x). \\
& \mathsf{if}\ \mathsf{mac}(\mathsf{proj}_1(x), k_M) = \mathsf{proj}_2(x)\ \mathsf{then} \\
& \quad \mathsf{if}\ n_T = \mathsf{proj}_1(\mathsf{sdec}(\mathsf{proj}_1(x), k_E))\ \mathsf{then}\ \ldots\ \mathsf{else} \\
& \quad \mathsf{out}(c, \mathrm{ERR\_nonce}) \\
& \mathsf{elseout}(c, \mathrm{ERR\_mac})
\end{array}
$$

Linkability issue :

$$\mathsf{new}\ k_E, k_M, k'_E, k'_M.\ T(k_E, k_M) \mid R(k_E, k_M) \not\approx T(k_E, k_M) \mid R(k'_E, k'_M)$$

Some general definitions

Let $I(\vec{k}, \vec{n})$ and $R(\vec{k}, \vec{n})$ be two roles of a protocol, where $\vec{k}$ represents identity parameters and $\vec{n}$ represent session parameters.

Definition

The protocol ensures strong unlinkability when:

$$! \text{ new } \vec{k}.\ ! \text{ new } \vec{n}.\ I(\vec{k}, \vec{n}) \mid R(\vec{k}, \vec{n}) \approx\ ! \text{ new } \vec{k}.\ \text{new } \vec{n}.\ I(\vec{k}, \vec{n}) \mid R(\vec{k}, \vec{n})$$

Definition

The protocol ensures anonymity when:

$$\mathcal{M} \approx \mathcal{M} \mid ! \text{ new } \vec{n}.\ I(\vec{\mathrm{k}_0}, \vec{n}) \mid R(\vec{\mathrm{k}_0}, \vec{n})$$

where $\mathcal{M}$ is the left process on the previous equivalence.

Observational equivalence

We write $P \Downarrow c$ when P can output on c after internal reductions, i.e. $P \leadsto^* \mathsf{out}(c, u).P' \mid P''$.

Definition

The binary relation $\mathcal{R}$ over closed processes is a observational bisimulation if it is symmetric and $P \,\mathcal{R}\, Q$ implies:

- for all c, $P \Downarrow c$ implies $Q \Downarrow c$;
- for all P', $P \leadsto^* P'$ implies $Q \leadsto^* \mathcal{R}\, P'$;
- for all R, $(P \mid R) \,\mathcal{R}\, (Q \mid R)$.

Observational equivalence is the largest observational bisimulation.

Observational equivalence

We write $P \Downarrow c$ when P can output on c after internal reductions, i.e. $P \leadsto^* \mathsf{out}(c, u).P' \mid P''$.

Definition

The binary relation $\mathcal{R}$ over closed processes is a observational bisimulation if it is symmetric and $P \, \mathcal{R} \, Q$ implies:

- for all c, $P \Downarrow c$ implies $Q \Downarrow c$;
- for all P', $P \leadsto^* P'$ implies $Q \leadsto^* \mathcal{R} \, P'$;
- for all R, $(P \mid R) \, \mathcal{R} (Q \mid R)$.

Observational equivalence is the largest observational bisimulation.

The quantification over all contexts makes it hard to prove obs. equiv, both by hand and mechanically.

Labelled bisimulation

Definition

The binary relation $\mathcal{R}$ over configurations is a bisimulation if it is symmetric and $A \mathcal{R} B$ implies:

- $\Phi(A) \sim \Phi(B)$;
- $A \xrightarrow{\tau} A'$ implies $B \xrightarrow{\tau}^* \mathcal{R} A'$;
- $A \xrightarrow{\alpha} A'$ implies $B \stackrel{\alpha}{\Rightarrow} \mathcal{R} A'$.

Bisimilarity is the largest bisimulation.

Theorem (Abadí, Blanchet & Fournet 2001/2017)

P and Q are observationally equivalent iff they are bisimilar.

Comparison with trace equivalence

Proposition

If A and B are bisimilar, then $A \approx B$.

Comparison with trace equivalence

Proposition

If A and B are bisimilar, then $A \approx B$.

Trace equivalence is a linear-time property, bisimularity is branching-time: trace equivalence does not "see" choice points.

Example

Assume a choice operator $P_1 + P_2 \xrightarrow{\tau} P_i$ for $i \in \{1, 2\}$.
$\mathsf{out}(a, \mathsf{ok}).(\mathsf{out}(b, \mathsf{ok}) + \mathsf{out}(c, \mathsf{ok})) \approx$
$\mathsf{out}(a, \mathsf{ok}).\mathsf{out}(b, \mathsf{ok}) + \mathsf{out}(a, \mathsf{ok}).\mathsf{out}(c, \mathsf{ok})$ but they are not bisimilar.

Comparison with trace equivalence

Proposition

If A and B are bisimilar, then $A \approx B$.

Trace equivalence is a linear-time property, bisimularity is branching-time: trace equivalence does not "see" choice points.

Example

Assume a choice operator $P_1 + P_2 \xrightarrow{\tau} P_i$ for $i \in \{1, 2\}$.
$\mathsf{out}(a, \mathsf{ok}).(\mathsf{out}(b, \mathsf{ok}) + \mathsf{out}(c, \mathsf{ok})) \approx$
$\mathsf{out}(a, \mathsf{ok}).\mathsf{out}(b, \mathsf{ok}) + \mathsf{out}(a, \mathsf{ok}).\mathsf{out}(c, \mathsf{ok})$ but they are not bisimilar.

Example without choice (Pous & Madiot)

Without choice, take two observably distinct actions α and β.
Consider $P := \alpha.(\alpha.(\alpha.\beta.\alpha|\beta.\beta)|\beta.\alpha)$ and $Q := \alpha.\beta.\alpha|\alpha.(\alpha.\beta.(\alpha|\beta)|\beta)$.
We have $P \approx Q$ but $P \xrightarrow{\alpha.\beta.\alpha} \alpha.\beta.\alpha|\beta.\beta|\alpha$ which cannot be matched by Q.

Comparison with trace equivalence

Proposition

If A and B are determinate, and $A \approx B$, then A and B are bisimilar.

Possible definitions of determinacy

A is determinate if, for all $A \stackrel{\mathsf{tr}}{\Rightarrow} A'$:

Comparison with trace equivalence

Proposition

If A and B are determinate, and $A \approx B$, then A and B are bisimilar.

Possible definitions of determinacy

A is determinate if, for all $A \overset{\text{tr}}{\Rightarrow} A'$:

- A' does not have two inputs (resp. outputs) on the same c at toplevel;

Comparison with trace equivalence

Proposition

If A and B are determinate, and $A \approx B$, then A and B are bisimilar.

Possible definitions of determinacy

A is determinate if, for all $A \stackrel{\mathsf{tr}}{\Rightarrow} A'$:

- A' does not have two inputs (resp. outputs) on the same c at toplevel;
- for all α, $A' \stackrel{\alpha}{\Rightarrow} A'_1$ and $A' \stackrel{\alpha}{\Rightarrow} A'_2$ imply $\Phi(A'_1) \sim \Phi(A'_2)$;

Comparison with trace equivalence

Proposition

If A and B are determinate, and $A \approx B$, then A and B are bisimilar.

Possible definitions of determinacy

A is determinate if, for all $A \stackrel{\mathsf{tr}}{\Rightarrow} A'$:

- A' does not have two inputs (resp. outputs) on the same c at toplevel;
- for all α, $A' \stackrel{\alpha}{\Rightarrow} A'_1$ and $A' \stackrel{\alpha}{\Rightarrow} A'_2$ imply $\Phi(A'_1) \sim \Phi(A'_2)$;
- for all α, $A' \stackrel{\alpha}{\Rightarrow} A'_1$ and $A' \stackrel{\alpha}{\Rightarrow} A'_2$ imply $A'_1 \approx A'_2$.

Bisimilarity in practice

The gap between bisim and trace equivalence (determinacy) may or may not matter depending on applications.

Bisimilarity is generally easier to prove than trace equivalence:

- by hand: bisimulation proof technique;
- mechanically: incrementally find matching processes.

In verification, even more constraining forms of equivalences are considered, e.g. diff-equivalence where the two processes must have the same structure and differ only in the terms that they use.

Tools

- diff-equivalence: proverif, tamarin (unbounded sessions)
- bisimilarity: SPEC (bounded sessions)
- trace equivalence: Apte/DeepSec, Akiss (bounded sessions)

Equivalence examples

Diff-equivalence successes

- Strong secrecy: $P[x := u]$ vs $P[x := 0]$?
- Anonymity: $P[x := A]$ vs $P[x := B]$?

Unlinkability: gray zone

- Not bisimilar in general, trace equiv. needed:

$$! \text{ new } k \; ! \text{ new } n, m.\; I(k, n) \mid R(k, m)$$

$$! \text{ new } k \text{ new } n, m.\; I(k, n) \mid R(k, m)$$

- Often diff-equivalent when no shared identity:

$$! \text{ new } k \; ! \text{ new } k' \text{new } n, m.\; I(k, n) \mid R(m)$$

$$! \text{ new} k \; ! \text{ new } k' \text{new } n, m.\; I(k', n) \mid R(m)$$

Summary

Static equivalence

- Indistinguishable sequences of messages
- Depends on equational theory, destructors vs. constructors

May testing & trace equivalence

- May testing: there exists an adversary (in the same model)
- Trace equivalence: the same traces can be observed
- Trace equivalence is a good approximation of may testing, often used in practice for verification.

Obs. equiv., bisimulation and diff-equiv.

- Obs. equiv = bisimulation = strongest "reasonable" equivalence
- Good properties: compositional, congruence, easier to check
- Common approximation for verification: diff-equivalence

A procedure for deciding static equivalence

for subterm-convergent equational theories

Outline

1. Definition of an (almost) finite set of equations $\mathsf{Eq}(\Phi)$ such that $\Phi \sim \Phi'$ iff $\Phi \models \mathsf{Eq}(\Phi')$ and $\Phi' \models \mathsf{Eq}(\Phi)$.
2. Effectiveness and complexity of computing and checking $\mathsf{Eq}(\Phi)$.

Subterm-convergent theories

Subterm-convergent theory

The theory E is subterm-convergent if there exists a rewriting relation $\rightarrow_{\mathrm{E}}$ generated from a finite set of pairs $\{ (l_i, r_i) \mid 1 \leq i \leq n \}$ such that:

- $\rightarrow_{\mathrm{E}}$ is convergent (terminating and confluent);
- $u =_{\mathrm{E}} v$ iff $u{\downarrow_{\mathrm{E}}} = v{\downarrow_{\mathrm{E}}}$;
- r_i is a constant or a subterm of l_i, for all $1 \leq i \leq n$.

Example

- Estd is subterm-convergent;
- theories for blind signatures, homomorphic encryptions, etc. are not.

Setup

Contexts

A context is a term built using Σ over special hole variables ($\_\_1$, $\_\_2 \dots$) and attacker names ($n, m, \dots \in \mathcal{N}_A$) which may not occur elsewhere. Context application is written $C[M_1, \dots, M_n]$.

Proposition

Let n be an (attacker) name. If $u =_{\mathsf{E}} v$ then $u[n := t] =_{\mathsf{E}} v[n := t]$.

Problem

Input: two frames Φ, Φ' of same domain
Output: $R_1\Phi =_{\mathsf{E}} R_2\Phi$ iff $R_1\Phi' =_{\mathsf{E}} R_2\Phi'$ for all $R_1, R_2 \in \mathcal{T}(\Sigma \cup \mathcal{N}_A, \mathcal{W})$?
Equivalently when $\Phi = \{ w_i \mapsto M_i \mid 1 \le i \le n \}$ (and similarly for Φ'):
$C_1[M_1, \dots, M_n] =_E C_2[M_1, \dots, M_n]$ iff
$C_1[M'_1, \dots, M'_n] =_E C_2[M'_1, \dots, M'_n]$ for all contexts C_1, C_2 with n holes ?

Definitions: sat(Φ) and Eq(Φ)

Set of messages sat(Φ)

Least subset of st(Φ) s.t. $M \in \mathrm{sat}(\Phi)$ in each of the following cases:

- $M = \Phi(w)$ for some $w \in \mathrm{dom}(\Phi)$;
- $M = f(M_1, \ldots, M_m)$ with $M_i \in \mathrm{sat}(\Phi)$ for all $i \in [1; m]$;
- $C[M_1, \ldots, M_m] \rightarrow_{\mathsf{E}} M$ with $|C| \leq c_{\mathsf{E}}$ and $M_i \in \mathrm{sat}(\Phi)$ for all i.

Proposition

For all $M \in \mathrm{sat}(\Phi)$ there exists a recipe R_M such that $R_M\Phi =_{\mathsf{E}} M$.

Set of formal equations Eq(Φ)

Contains all $C_1[R_{M_1}, \ldots, R_{M_m}] \doteq C_2[R_{N_1}, \ldots, R_{N_n}]$ such that

- $(C_1[R_{M_1}, \ldots, R_{M_m}] =_{\mathsf{E}} C_2[R_{N_1}, \ldots, R_{N_n}])\Phi$;
- $M_i \in \mathrm{sat}(\Phi)$ for all i, $N_j \in \mathrm{sat}(\Phi)$ for all j;
- $|C_1|, |C_2| \leq c_{\mathsf{E}}$.

Main results

Theorem

$\Phi \sim \Phi'$ iff $\Phi \models \mathrm{Eq}(\Phi')$ and $\Phi' \models \mathrm{Eq}(\Phi)$.

Let Φ and Φ' be two frames such that $\Phi' \models \mathrm{Eq}(\Phi)$. We have:

Lemma

Let C_1 and C_2 be contexts and $\{M_1, \ldots, M_m, N_1, \ldots, N_n\} \subseteq \mathrm{sat}(\Phi)$.
If $C_1[M_i]_i = C_2[N_j]_j$ then $(C_1[R_{M_i}]_i =_{\mathsf{E}} C_2[R_{N_j}]_j)\Phi'$.

Lemma

Let C_1 be a context and $\{M_1, \ldots, M_m\} \subseteq \mathrm{sat}(\Phi)$.
If $C_1[M_i]_i \to_{\mathsf{E}}^* T$ there exist C_2 and $\{N_1, \ldots, N_n\} \subseteq \mathrm{sat}(\Phi)$ such that $C_2[N_j]_j = T$ and $(C_1[M_i]_i =_{\mathsf{E}} C_2[N_j]_j)\Phi'$.

Procedure

The proofs work with $c_{\mathsf{E}} = \max(1 + \mathsf{arity}(\Sigma), \max_i(|l_i|))$.

Effectiveness

Checking $\Phi \sim \Phi'$ boils down to check $\Phi' \models \mathsf{Eq}(\Phi)$ (and conversely):

- The least fixed point $\mathsf{sat}(\Phi) \subseteq \mathsf{st}(\Phi)$ can be computed.

Procedure

The proofs work with $c_{\mathsf{E}} = \max(1 + \mathsf{arity}(\Sigma), \max_i(|l_i|))$.

Effectiveness

Checking $\Phi \sim \Phi'$ boils down to check $\Phi' \models \mathsf{Eq}(\Phi)$ (and conversely):

- The least fixed point $\mathsf{sat}(\Phi) \subseteq \mathsf{st}(\Phi)$ can be computed.
- The set $\mathsf{Eq}(\Phi)$ may be infinite due to attacker names, but checking $\Phi' \models \mathsf{Eq}(\Phi)$ can be done by considering only $2c_{\mathsf{E}}$ names.

Procedure

The proofs work with $c_{\mathsf{E}} = \max(1 + \mathrm{arity}(\Sigma), \max_i(|l_i|))$.

Effectiveness

Checking $\Phi \sim \Phi'$ boils down to check $\Phi' \models \mathrm{Eq}(\Phi)$ (and conversely):

- The least fixed point $\mathrm{sat}(\Phi) \subseteq \mathrm{st}(\Phi)$ can be computed.
- The set $\mathrm{Eq}(\Phi)$ may be infinite due to attacker names, but checking $\Phi' \models \mathrm{Eq}(\Phi)$ can be done by considering only $2c_{\mathsf{E}}$ names.

Complexity: PTIME($|\Phi|$)

- Using a DAG representation with maximum sharing, $\mathrm{sat}(\Phi)$ can be computed in PTIME, and is of polynomial size.

Procedure

The proofs work with $c_{\mathsf{E}} = \max(1 + \mathsf{arity}(\Sigma), \max_i(|l_i|))$.

Effectiveness

Checking $\Phi \sim \Phi'$ boils down to check $\Phi' \models \mathsf{Eq}(\Phi)$ (and conversely):

- The least fixed point $\mathsf{sat}(\Phi) \subseteq \mathsf{st}(\Phi)$ can be computed.
- The set $\mathsf{Eq}(\Phi)$ may be infinite due to attacker names, but checking $\Phi' \models \mathsf{Eq}(\Phi)$ can be done by considering only $2c_{\mathsf{E}}$ names.

Complexity: PTIME($|\Phi|$)

- Using a DAG representation with maximum sharing, $\mathsf{sat}(\Phi)$ can be computed in PTIME, and is of polynomial size.
- We have $|\mathsf{Eq}(\Phi)| \leq (|\Sigma| + 2c_{\mathsf{E}} + |\Phi|)^{2c_{\mathsf{E}}}$
  and each equation can be checked in polynomial time.

Concluding remarks

Static equivalence in practice

- The procedure is not practical, and subterm-convergence is restrictive.
- Other approaches fare better, e.g. Horn-clauses in (A)KISS.

Concluding remarks

Static equivalence in practice

- The procedure is not practical, and subterm-convergence is restrictive.
- Other approaches fare better, e.g. Horn-clauses in (A)KISS.

From frame to protocol equivalences

Bounded case:

- Bisimulation can be reduced to symbolic equivalence, i.e. static equivalence with unknowns subject to deducibility.
- Trace equivalence reduces to symb. equiv. over sets of constraints, which can be solved through simplification rules.

Unbounded case:

- Proverif and Tamarin allow to (approximately) verify diff-equivalence.
- We'll cover Proverif's saturation-based approach.