Symbolic Verification of Cryptographic Protocols

Protocol Equivalences

David Baelde

LSV, ENS Paris-Saclay

2018–2019

Static equivalence

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

When are two sequences of messages 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 messages 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 messages 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 messages 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 messages 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

As before, consider frames in $\mathcal{N}^* \times (\mathcal{W} \to \mathcal{T}_c(\mathcal{N}))$:
1^{st} component = bound/private names, noted $\mathrm{bn}(\Phi)$;
2^{nd} component = intruder's knowledge, addressed via handles of $\mathrm{dom}(\Phi)$.

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 $M \in \mathcal{T}(\mathcal{W} \cup \mathcal{N} \setminus \mathrm{bn}(\Phi_1, \Phi_2))$, $M\Phi_1\Downarrow$ iff $M\Phi_2\Downarrow$;
- for all $M, N \in \mathcal{T}(\mathcal{W} \cup \mathcal{N} \setminus \mathrm{bn}(\Phi_1, \Phi_2))$,
  $M\Phi_1\Downarrow =_{\mathsf{E}} N\Phi_1\Downarrow$ iff $M\Phi_2\Downarrow =_{\mathsf{E}} N\Phi_2\Downarrow$.

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 have only constructors and
the standard equations for pairs and (a)symmetric encryption.

Examples (bis)

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

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.

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

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 $\mathsf{senc}(\mathsf{sdec}(x, y), y) = x$... and no getkey primitive for aenc.

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

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

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.

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 !

Trace equivalence

Weak labelled transitions

We write $A \overset{\mathsf{tr}}{\Rightarrow} B$ 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

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

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 $\Phi(A) \sim \Phi(B)$, but not $\Phi(A) = \Phi(B)$.
- The definition really makes sense only when $\mathrm{bn}(\Phi(A)) = \mathrm{bn}(\Phi(B))$.
- 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...*

Trace equivalence $\subseteq$ may-testing ?

Proposition

If $(P, \emptyset) \approx (Q, \emptyset)$ *then they are in may-testing equivalence. . .*
provided computation is deterministic, i.e.
for all t, u *and* v *such that* $t \Downarrow u$, *we have* $t \Downarrow v$ *iff* $u =_{\mathsf{E}} v$.

Trace equivalence $\subseteq$ may-testing ?

Proposition

If $(P, \emptyset) \approx (Q, \emptyset)$ then they are in may-testing equivalence...
provided computation is deterministic, i.e.
for all t, u and v such that $t \Downarrow u$, we have $t \Downarrow v$ iff $u =_{\mathsf{E}} v$.

Proof idea.

Decompose $P \mid T \leadsto^* \mathsf{out}(\mathbb{T}, \_) \mid \_$ into internal reductions of P and T, and communications between the two. This yields a trace of P, which Q can simulate. Compose this with the reductions of T to obtain $Q \mid T \leadsto^* \mathsf{out}(\mathbb{T}, \_) \mid \_$. $\square$

Devil is in the details! there is a counter-example when computation is non-deterministic because traces do not keep track of how recipes are evaluated.

May testing $\subseteq$ trace equivalence ?

Proposition

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

May testing $\subseteq$ trace equivalence ?

Proposition

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

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

and similarly for Q.

May testing $\subseteq$ trace equivalence ?

Proposition

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

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

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 $\subseteq$ 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.

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

Anonymity

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

Application: private authentication

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

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

Anonymity

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

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:

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

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. $P \rightsquigarrow^* \mathsf{out}(c, u).P' \mid P''$.

Definition

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

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

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. $P \rightsquigarrow^* \mathrm{out}(c, u).P' \mid P''$.

Definition

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

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

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 $A \mathrel{\mathcal{R}} B$ implies:

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

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 $i \in \{1, 2\}$.
$\mathrm{out}(a, \mathrm{ok}).(\mathrm{out}(b, \mathrm{ok}) + \mathrm{out}(c, \mathrm{ok})) \approx$
$\mathrm{out}(a, \mathrm{ok}).\mathrm{out}(b, \mathrm{ok}) + \mathrm{out}(a, \mathrm{ok}).\mathrm{out}(c, \mathrm{ok})$ 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.

Definition (A possible definition of determinacy)

A is determinate if, for all $A \overset{\mathrm{tr}}{\Rightarrow} A'$: A' does not have two inputs (resp. outputs) on the same c at toplevel.

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