Modelling and verifying privacy-type properties in applied-pi calculus

Stéphanie Delaune

Post-doctoral student at LORIA – Cassis Project

Thursday 26th April

Electronic voting

Advantages:

- Convenient,
- Efficient facilities for tallying votes.

Drawbacks:

- Risk of large-scale and undetectable fraud,
- Such protocols are extremely error-prone.

> *"A 15-year-old in a garage could manufacture smart cards and sell them on the Internet that would allow for multiple votes"* Avi Rubin

Possible issue: formal methods

abstract analysis of the protocol against formally-stated properties

Electronic voting

Advantages:

- Convenient,
- Efficient facilities for tallying votes.

Drawbacks:

- Risk of large-scale and undetectable fraud,
- Such protocols are extremely error-prone.

> *"A 15-year-old in a garage could manufacture smart cards and sell them on the Internet that would allow for multiple votes"* Avi Rubin

Possible issue: formal methods

abstract analysis of the protocol against formally-stated properties

Example: Fujioka *et al.* protocol (1992)

Cryptographic primitives as an equational theory

- Public Key

$$\mathsf{getpk}(\mathsf{host}(\mathsf{pubkey})) = \mathsf{pubkey}$$

- Commitment

$$\mathsf{open}(\mathsf{commit}(\mathsf{m}, \mathsf{r}), \mathsf{r}) = \mathsf{m}$$

- Blind Signature

$$\begin{aligned}
\mathsf{checksign}(\mathsf{sign}(\mathsf{m}, \mathsf{sk}), \mathsf{pk}(\mathsf{sk})) &= \mathsf{m} \\
\mathsf{unblind}(\mathsf{blind}(\mathsf{m}, \mathsf{r}), \mathsf{r}) &= \mathsf{m} \\
\mathsf{unblind}(\mathsf{sign}(\mathsf{blind}(\mathsf{m}, \mathsf{r}), \mathsf{sk}), \mathsf{r}) &= \mathsf{sign}(\mathsf{m}, \mathsf{sk})
\end{aligned}$$

Example: Fujioka *et al.* protocol (1992)

First Phase:

the voter gets a "token" from the administrator.

1. $V \rightarrow A \quad : \quad V, sign(blind(commit(vote, r), b), V)$
2. $A \rightarrow V \quad : \quad sign(blind(commit(vote, r), b), A)$

$\longrightarrow$ to ensure privacy, blind signatures are used

Voting phase:

3. $V \rightarrow C \quad : \quad commit(vote, r), \; sign(commit(vote, r), A)$
4. $C \rightarrow \quad : \quad l, \; commit(vote, r), \; sign(commit(vote, r), A)$

Counting phase:

5. $V \rightarrow C \quad : \quad l, r$
6. C publishes the outcome of the vote

$\longrightarrow$ to ensure privacy, anonymous channel are used at step 3 and 5

Example: Fujioka *et al.* protocol (1992)

First Phase:

the voter gets a "token" from the administrator.

1. $V \rightarrow A$: $V, sign(blind(commit(vote, r), b), V)$
2. $A \rightarrow V$: $sign(blind(commit(vote, r), b), A)$

$\longrightarrow$ to ensure privacy, blind signatures are used

Voting phase:

3. $V \rightarrow C$: $commit(vote, r),\ sign(commit(vote, r), A)$
4. $C \rightarrow$: $l,\ commit(vote, r),\ sign(commit(vote, r), A)$

Counting phase:

5. $V \rightarrow C$: l, r
6. C publishes the outcome of the vote

$\longrightarrow$ to ensure privacy, anonymous channel are used at step 3 and 5

Example: Fujioka *et al.* protocol (1992)

First Phase:

the voter gets a "token" from the administrator.

1. $V \rightarrow A \quad : \quad V, sign(blind(commit(vote, r), b), V)$
2. $A \rightarrow V \quad : \quad sign(blind(commit(vote, r), b), A)$

$\longrightarrow$ to ensure privacy, blind signatures are used

Voting phase:

3. $V \rightarrow C \quad : \quad commit(vote, r), \; sign(commit(vote, r), A)$
4. $C \rightarrow \quad : \quad l, \; commit(vote, r), \; sign(commit(vote, r), A)$

Counting phase:

5. $V \rightarrow C \quad : \quad l, r$
6. C publishes the outcome of the vote

$\longrightarrow$ to ensure privacy, anonymous channel are used at step 3 and 5

Eligibility: only legitimate voters can vote, and only once

Fairness: no early results can be obtained which could influence the remaining voters

Individual verifiability:
a voter can verify that her vote was really counted

Universal verifiability:
the published outcome really is the sum of all the votes

Belgique - Election 2004 - http://www.poureva.be/ - (C) Kanar

Privacy-type security properties

Privacy: the fact that a particular voted in a particular way is not revealed to anyone

Receipt-freeness: a voter cannot prove that she voted in a certain way (this is important to protect voters from coercion)

Coercion-resistance: same as receipt-freeness, but the coercer interacts with the voter during the protocol, (*e.g.* by preparing messages)

Summary

Observations:

- Definitions of security properties are often insufficiently precise
- No clear distinction between receipt-freeness and coercion-resistance

Goal:

1. Propose "formal methods" definitions of privacy-type properties,
2. Design automatic procedures to verify them.

Difficulties:

- equivalence based-security properties are harder than reachability properties (*e.g.* secrecy, authentication),
- electronic voting protocols are often more complex than authentication protocols,
- less classical cryptographic primitives (*e.g.* blind signature).

Summary

Observations:

- Definitions of security properties are often insufficiently precise
- No clear distinction between receipt-freeness and coercion-resistance

Goal:

1. Propose "formal methods" definitions of privacy-type properties,
2. Design automatic procedures to verify them.

Difficulties:

- equivalence based-security properties are harder than reachability properties (*e.g.* secrecy, authentication),
- electronic voting protocols are often more complex than authentication protocols,
- less classical cryptographic primitives (*e.g.* blind signature).

Summary

Observations:

- Definitions of security properties are often insufficiently precise
- No clear distinction between receipt-freeness and coercion-resistance

Goal:

1. Propose "formal methods" definitions of privacy-type properties,
2. Design automatic procedures to verify them.

Difficulties:

- equivalence based-security properties are harder than reachability properties (*e.g.* secrecy, authentication),
- electronic voting protocols are often more complex than authentication protocols,
- less classical cryptographic primitives (*e.g.* blind signature).

Results and Work in Progress

Modelling:

- Formalisation of privacy, receipt-freeness and coercion-resistance as some kind of observational equivalence in the applied pi-calculus,
- Coercion-Resistance ⇒ Receipt-Freeness ⇒ Privacy,

Case Studies:

- Fujioka *et al.*'92 – commitment and blind signature,
- Okamoto'96 – trap-door bit commitment and blind signature,
- Lee *et al.*'03 – re-encryption and designated verifier proof of re-encryption.

Verification: How to check such privacy-type properties?

- by using an existing tool (*e.g.* ProVerif)
- by developping new techniques (symbolic bisimulation)

Results and Work in Progress

Modelling:

- Formalisation of privacy, receipt-freeness and coercion-resistance as some kind of observational equivalence in the applied pi-calculus,
- Coercion-Resistance $\Rightarrow$ Receipt-Freeness $\Rightarrow$ Privacy,

Case Studies:

- Fujioka *et al.*'92 – commitment and blind signature,
- Okamoto'96 – trap-door bit commitment and blind signature,
- Lee *et al.*'03 – re-encryption and designated verifier proof of re-encryption.

Verification: How to check such privacy-type properties?

- by using an existing tool (*e.g.* ProVerif)
- by developping new techniques (symbolic bisimulation)

Results and Work in Progress

Modelling:

- Formalisation of privacy, receipt-freeness and coercion-resistance as some kind of observational equivalence in the applied pi-calculus,
- Coercion-Resistance $\Rightarrow$ Receipt-Freeness $\Rightarrow$ Privacy,

Case Studies:

- Fujioka *et al.*'92 – commitment and blind signature,
- Okamoto'96 – trap-door bit commitment and blind signature,
- Lee *et al.*'03 – re-encryption and designated verifier proof of re-encryption.

Verification: How to check such privacy-type properties?

- by using an existing tool (*e.g.* ProVerif)
- by developping new techniques (symbolic bisimulation)

Outline of the talk

1. Introduction
2. Applied π-calculus
3. Formalisation of Privacy-type Properties (Privacy, Receipt-Freeness)
4. Verification of privacy-type properties (works in progress)
5. Conclusion and Future Works

Outline of the talk

1. Introduction
2. Applied π-calculus
3. Formalisation of Privacy-type Properties (Privacy, Receipt-Freeness)
4. Verification of privacy-type properties (works in progress)
5. Conclusion and Future Works

Voting protocols in the applied π-calculus

Definition (Voting process)

$$VP \equiv \nu\tilde{n}.(V\sigma_1 \mid \cdots \mid V\sigma_n \mid A_1 \mid \cdots \mid A_m)$$

- $V\sigma_i$: voter processes and $v \in dom(\sigma_i)$ refers to the value of the vote
- A_j: election authorities which are required to be honest,
- $\tilde{n}$: channel names

$\hookrightarrow$ S is a context which is as VP but has a hole instead of two of the $V\sigma_i$

Example: Fujioka *et al.* (1992)

Main Process

```
process
   (* private channels *)
   ν. privCh; ν. pkaCh1; ν. pkaCh2; ν. skaCh;
   ν. skvaCh; ν. skvbCh;
   (* administrators *)
   (processK | processA | processA | processC | processC |
   (* voters *)
   (let skvCh = skvaCh in let v = a in processV) |
   (let skvCh = skvbCh in let v = b in processV) )
```

Example: Fujioka *et al.* (1992)

```
let processV =
   (* his private key *)
   in(skvCh,skv); let hostv = host(pk(skv)) in
   (* public keys of the administrator *)
   in(pkaCh1,pubka);
   ν. blinder; ν. r; let committedvote = commit(v,r) in
   let blindedcommittedvote=blind(committedvote,blinder) in
   out(ch,(hostv,sign(blindedcommittedvote,skv)));

   in(ch,m2);
   let result = checksign(m2,pubka) in
   if result = blindedcommittedvote then
   let signedcommittedvote=unblind(m2,blinder) in
   phase 1;
   out(ch,(committedvote,signedcommittedvote));
   in(ch,(l,=committedvote,=signedcommittedvote));
   phase 2;
   out(ch,(l,r)).
```

Example: Fujioka *et al.* (1992)

```
let processV =
   (* his private key *)
   in(skvCh,skv); let hostv = host(pk(skv)) in
   (* public keys of the administrator *)
   in(pkaCh1,pubka);
   ν. blinder; ν. r; let committedvote = commit(v,r) in
   let blindedcommittedvote=blind(committedvote,blinder) in
   out(ch,(hostv,sign(blindedcommittedvote,skv)));
   in(ch,m2);
   let result = checksign(m2,pubka) in
   if result = blindedcommittedvote then
   let signedcommittedvote=unblind(m2,blinder) in
   phase 1;
   out(ch,(committedvote,signedcommittedvote));
   in(ch,(l,=committedvote,=signedcommittedvote));
   phase 2;
   out(ch,(l,r)).
```

Example: Fujioka *et al.* (1992)

```
let processV =
   (* his private key *)
   in(skvCh,skv); let hostv = host(pk(skv)) in
   (* public keys of the administrator *)
   in(pkaCh1,pubka);
   ν. blinder; ν. r; let committedvote = commit(v,r) in
   let blindedcommittedvote=blind(committedvote,blinder) in
   out(ch,(hostv,sign(blindedcommittedvote,skv)));
   in(ch,m2);
   let result = checksign(m2,pubka) in
   if result = blindedcommittedvote then
   let signedcommittedvote=unblind(m2,blinder) in
   phase 1;
   out(ch,(committedvote,signedcommittedvote));
   in(ch,(l,=committedvote,=signedcommittedvote));
   phase 2;
   out(ch,(l,r)).
```

Example: Fujioka *et al.* (1992)

```
let processV =
   (* his private key *)
   in(skvCh,skv); let hostv = host(pk(skv)) in
   (* public keys of the administrator *)
   in(pkaCh1,pubka);
   ν. blinder; ν. r; let committedvote = commit(v,r) in
   let blindedcommittedvote=blind(committedvote,blinder) in
   out(ch,(hostv,sign(blindedcommittedvote,skv)));
   in(ch,m2);
   let result = checksign(m2,pubka) in
   if result = blindedcommittedvote then
   let signedcommittedvote=unblind(m2,blinder) in
   phase 1;
   out(ch,(committedvote,signedcommittedvote));
   in(ch,(l,=committedvote,=signedcommittedvote));
   phase 2;
   out(ch,(l,r)).
```

Observational equivalence

Observational equivalence ($\approx$)

The largest symmetric relation $\mathcal{R}$ on processes such that $A \; \mathcal{R} \; B$ implies

1. if $A \Downarrow a$, then $B \Downarrow a$,
2. if $A \rightarrow^* A'$, then $B \rightarrow^* B'$ and $A' \; \mathcal{R} \; B'$ for some B',
3. $C[A] \; \mathcal{R} \; C[B]$ for all closing evaluation contexts $C[\,]$.

$\longrightarrow$ $A \Downarrow a$ when A can send a message on the channel a.

Observational equivalence

Observational equivalence ($\approx$)

The largest symmetric relation $\mathcal{R}$ on processes such that $A \ \mathcal{R} \ B$ implies

1. if $A \Downarrow a$, then $B \Downarrow a$,
2. if $A \rightarrow^* A'$, then $B \rightarrow^* B'$ and $A' \ \mathcal{R} \ B'$ for some B',
3. $C[A] \ \mathcal{R} \ C[B]$ for all closing evaluation contexts $C[\,]$.

$\longrightarrow$ $A \Downarrow a$ when A can send a message on the channel a.

Example 1: $\mathrm{out}(a, s) \not\approx \mathrm{out}(a, s')$

$\longrightarrow$ $C[\_] = \mathrm{in}(a, x).\mathrm{if}\ x = s\ \mathrm{then}\ \mathrm{out}(c, ok) \mid \_$

Observational equivalence

Observational equivalence ($\approx$)

The largest symmetric relation $\mathcal{R}$ on processes such that $A \mathrel{\mathcal{R}} B$ implies

1. if $A \Downarrow a$, then $B \Downarrow a$,
2. if $A \rightarrow^* A'$, then $B \rightarrow^* B'$ and $A' \mathrel{\mathcal{R}} B'$ for some B',
3. $C[A] \mathrel{\mathcal{R}} C[B]$ for all closing evaluation contexts $C[\,]$.

$\longrightarrow$ $A \Downarrow a$ when A can send a message on the channel a.

Example 2:

$$\nu s.\mathrm{out}(a, \mathrm{enc}(s, k)).\mathrm{out}(a, \mathrm{enc}(s, k'))$$
$$\not\approx$$
$$\nu s, s'.\mathrm{out}(a, \mathrm{enc}(s, k)).\mathrm{out}(a, \mathrm{enc}(s', k'))$$

$\longrightarrow C[\_] = \mathrm{in}(a, x).\mathrm{in}(a, y).\mathrm{if}\ (\mathrm{dec}(x, k) = \mathrm{dec}(y, k'))\ \mathrm{then}\ \mathrm{out}(c, ok) \mid \_$

Observational equivalence ($\approx$)

The largest symmetric relation $\mathcal{R}$ on processes such that $A \mathrel{\mathcal{R}} B$ implies

1. if $A \Downarrow a$, then $B \Downarrow a$,
2. if $A \rightarrow^* A'$, then $B \rightarrow^* B'$ and $A' \mathrel{\mathcal{R}} B'$ for some B',
3. $C[A] \mathrel{\mathcal{R}} C[B]$ for all closing evaluation contexts $C[\,]$.

$\longrightarrow$ $A \Downarrow a$ when A can send a message on the channel a.

Example 3: $\nu s.\mathrm{out}(a, s) \approx \nu s.\mathrm{out}(a, h(s))$

Labeled bisimilarity

Labeled bisimilarity ($\approx_\ell$)

The largest symmetric relation $\mathcal{R}$ on closed extended processes, such that $A \mathrel{\mathcal{R}} B$ implies

1. $\phi(A) \approx_s \phi(B)$ (static equivalence)
2. if $A \rightarrow A'$, then $B \rightarrow^* B'$ and $A' \mathrel{\mathcal{R}} B'$ for some B',
3. if $A \xrightarrow{\alpha} A'$, then $B \rightarrow^* \xrightarrow{\alpha} \rightarrow^* B'$ and $A' \mathrel{\mathcal{R}} B'$ for some B'.

Theorem ; [Abadi & Fournet, 01]

Observational equivalence is labeled bisimilarity: $A \approx B \;\Leftrightarrow\; A \approx_\ell B$.

Static equivalence

A frame is a process of the form $\nu\tilde{n}.(\{^{M_1}/_{x_1}\} \mid \ldots \mid \{^{M_n}/_{x_n}\})$.

Static equivalence ($\approx_s$)

Let $\phi_1 = \nu\tilde{n}_1.\sigma_1$ and $\phi_2 = \nu\tilde{n}_2.\sigma_2$ be two frames. We have that $\phi_1 \approx_s \phi_2$ when

- $dom(\phi_1) = dom(\phi_2)$
- for all terms U, V such that $(fn(U) \cup fn(V)) \cap (\tilde{n}_1 \cup \tilde{n}_2) = \emptyset$,

$$(U =_E V)\sigma_1 \text{ iff } (U =_E V)\sigma_2$$

Static equivalence

A frame is a process of the form $\nu\tilde{n}.(\{^{M_1}/_{x_1}\} \mid \ldots \mid \{^{M_n}/_{x_n}\})$.

Static equivalence ($\approx_s$)

Let $\phi_1 = \nu\tilde{n}_1.\sigma_1$ and $\phi_2 = \nu\tilde{n}_2.\sigma_2$ be two frames. We have that $\phi_1 \approx_s \phi_2$ when

- $dom(\phi_1) = dom(\phi_2)$
- for all terms U, V such that $(fn(U) \cup fn(V)) \cap (\tilde{n}_1 \cup \tilde{n}_2) = \emptyset$,

$$(U =_E V)\sigma_1 \text{ iff } (U =_E V)\sigma_2$$

Example 1: $\nu k.(\{^{enc(a,k)}/_x\} \mid \{^k/_y\}) \not\approx_s \nu k.(\{^{enc(b,k)}/_x\} \mid \{^k/_y\})$

$\longrightarrow \quad (U, V) = (\mathrm{dec}(x, y), a)$

Static equivalence

A frame is a process of the form $\nu\tilde{n}.(\{^{M_1}/_{x_1}\} \mid \ldots \mid \{^{M_n}/_{x_n}\})$.

Static equivalence ($\approx_s$)

Let $\phi_1 = \nu\tilde{n}_1.\sigma_1$ and $\phi_2 = \nu\tilde{n}_2.\sigma_2$ be two frames. We have that $\phi_1 \approx_s \phi_2$ when

- $dom(\phi_1) = dom(\phi_2)$
- for all terms U, V such that $(fn(U) \cup fn(V)) \cap (\tilde{n}_1 \cup \tilde{n}_2) = \emptyset$,

$$(U =_E V)\sigma_1 \text{ iff } (U =_E V)\sigma_2$$

Example 2: $\nu k, a.(\{^{enc(a,k)}/_x\} \mid \{^k/_y\}) \approx_s \nu k, b.(\{^{enc(b,k)}/_x\} \mid \{^k/_y\})$

Static equivalence

A frame is a process of the form $\nu\tilde{n}.(\{^{M_1}/_{x_1}\} \mid \ldots \mid \{^{M_n}/_{x_n}\})$.

Static equivalence ($\approx_s$)

Let $\phi_1 = \nu\tilde{n}_1.\sigma_1$ and $\phi_2 = \nu\tilde{n}_2.\sigma_2$ be two frames. We have that $\phi_1 \approx_s \phi_2$ when

- $dom(\phi_1) = dom(\phi_2)$
- for all terms U, V such that $(fn(U) \cup fn(V)) \cap (\tilde{n}_1 \cup \tilde{n}_2) = \emptyset$,

$$(U =_E V)\sigma_1 \text{ iff } (U =_E V)\sigma_2$$

Example 3: $\nu k.\{^{enc(a,k)}/_x\} \approx_s \nu k.\{^{enc(b,k)}/_x\}$

Outline of the talk

Formalisation of privacy

Classically modeled as observational equivalences between two slightly different processes P_1 and P_2, but

- changing the identity does not work, as identities are revealed
- changing the vote does not work, as the votes are revealed at the end

Solution:
$\hookrightarrow$ consider 2 honest voters and swap their votes

A voting protocol respects privacy if

$$S[V_A\{^a/_v\} \mid V_B\{^b/_v\}] \approx S[V_A\{^b/_v\} \mid V_B\{^a/_v\}].$$

Formalisation of privacy

Classically modeled as observational equivalences between two slightly different processes P_1 and P_2, but

- changing the identity does not work, as identities are revealed
- changing the vote does not work, as the votes are revealed at the end

Solution:
$\hookrightarrow$ consider 2 honest voters and swap their votes

A voting protocol respects privacy if

$$S[V_A\{^a/_v\} \mid V_B\{^b/_v\}] \approx S[V_A\{^b/_v\} \mid V_B\{^a/_v\}].$$

Naive example 1

Voter process

$$V = \mathsf{out}(ch, \{v\}_{\mathsf{pub}(s)})$$

What about privacy?

$$V_A\{^{a}/_{v}\} \mid V_B\{^{b}/_{v}\} \overset{?}{\approx} V_A\{^{b}/_{v}\} \mid V_B\{^{a}/_{v}\}$$

i.e.

$$\mathsf{out}(ch, \{a\}_{\mathsf{pub}(s)}) \mid \mathsf{out}(ch, \{b\}_{\mathsf{pub}(s)}) \overset{?}{\approx} \mathsf{out}(ch, \{b\}_{\mathsf{pub}(s)}) \mid \mathsf{out}(ch, \{a\}_{\mathsf{pub}(s)})$$

$\longrightarrow$ OK

Naive example 1

Voter process

$$V = \mathsf{out}(ch, \{v\}_{\mathsf{pub}(s)})$$

What about privacy?

$$V_A\{^a/_v\} \mid V_B\{^b/_v\} \overset{?}{\approx} V_A\{^b/_v\} \mid V_B\{^a/_v\}$$

i.e.

$$\mathsf{out}(ch, \{a\}_{\mathsf{pub}(s)}) \mid \mathsf{out}(ch, \{b\}_{\mathsf{pub}(s)}) \overset{?}{\approx} \mathsf{out}(ch, \{b\}_{\mathsf{pub}(s)}) \mid \mathsf{out}(ch, \{a\}_{\mathsf{pub}(s)})$$

$\longrightarrow$ OK

Naive example 1

Voter process

$$V = \text{out}(ch, \{v\}_{\text{pub}(s)})$$

What about privacy?

$$V_A\{^a/_v\} \mid V_B\{^b/_v\} \stackrel{?}{\approx} V_A\{^b/_v\} \mid V_B\{^a/_v\}$$

i.e.

$$\text{out}(ch, \{a\}_{\text{pub}(s)}) \mid \text{out}(ch, \{b\}_{\text{pub}(s)}) \stackrel{?}{\approx} \text{out}(ch, \{b\}_{\text{pub}(s)}) \mid \text{out}(ch, \{a\}_{\text{pub}(s)})$$

$\longrightarrow$ OK

Naive example 2

Voter process

$$V(Id) = \mathrm{out}(ch, \langle Id, \{v\}_{\mathrm{pub}(s)}\rangle)$$

What about privacy?

$$V_A\{^a/_v\} \mid V_B\{^b/_v\} \stackrel{?}{\approx} V_A\{^b/_v\} \mid V_B\{^a/_v\}$$

i.e.

$$\mathrm{out}(ch, \langle A, \{a\}_{\mathrm{pub}(s)}\rangle) \mid \mathrm{out}(ch, \langle B, \{b\}_{\mathrm{pub}(s)}\rangle)$$
$$\stackrel{?}{\approx}$$
$$\mathrm{out}(ch, \langle A, \{b\}_{\mathrm{pub}(s)}\rangle) \mid \mathrm{out}(ch, \langle B, \{a\}_{\mathrm{pub}(s)}\rangle)$$

$\longrightarrow$ NOT OK (with deterministic encryption)
However, if we consider probabilistic encryption, then privacy holds.

Naive example 2

Voter process

$$V(Id) = \mathsf{out}(ch, \langle Id, \{v\}_{\mathsf{pub}(s)}\rangle)$$

What about privacy?

$$V_A\{^a/_v\} \mid V_B\{^b/_v\} \overset{?}{\approx} V_A\{^b/_v\} \mid V_B\{^a/_v\}$$

i.e.

$$\mathsf{out}(ch, \langle A, \{a\}_{\mathsf{pub}(s)}\rangle) \mid \mathsf{out}(ch, \langle B, \{b\}_{\mathsf{pub}(s)}\rangle)$$
$$\overset{?}{\approx}$$
$$\mathsf{out}(ch, \langle A, \{b\}_{\mathsf{pub}(s)}\rangle) \mid \mathsf{out}(ch, \langle B, \{a\}_{\mathsf{pub}(s)}\rangle)$$

$\longrightarrow$ NOT OK (with deterministic encryption)
However, if we consider probabilistic encryption, then privacy holds.

Naive example 2

Voter process

$$V(Id) = \text{out}(ch, \langle Id, \{v\}_{\text{pub}(s)} \rangle)$$

What about privacy?

$$V_A\{^a/_v\} \mid V_B\{^b/_v\} \overset{?}{\approx} V_A\{^b/_v\} \mid V_B\{^a/_v\}$$

i.e.

$$\text{out}(ch, \langle A, \{a\}_{\text{pub}(s)} \rangle) \mid \text{out}(ch, \langle B, \{b\}_{\text{pub}(s)} \rangle)$$
$$\overset{?}{\approx}$$
$$\text{out}(ch, \langle A, \{b\}_{\text{pub}(s)} \rangle) \mid \text{out}(ch, \langle B, \{a\}_{\text{pub}(s)} \rangle)$$

$\longrightarrow$ NOT OK (with deterministic encryption)
However, if we consider probabilistic encryption, then privacy holds.

Example: Fujioka *et al.* protocol (1992)

First Phase:
the voter gets a "token" from the administrator.

1. $V \rightarrow A \quad : \quad V, sign(blind(commit(vote, r), b), V)$
2. $A \rightarrow V \quad : \quad sign(blind(commit(vote, r), b), A)$

Voting phase:

3. $V \rightarrow C \quad : \quad commit(vote, r),\ sign(commit(vote, r), A)$
4. $C \rightarrow \quad : \quad l,\ commit(vote, r),\ sign(commit(vote, r), A)$

Counting phase:

5. $V \rightarrow C \quad : \quad l, r$
6. C publishes the outcome of the vote

What about privacy?

$$\nu \text{pkaCh1}.(V_A\{^a/_v\} \mid V_B\{^b/_v\} \mid \text{processK}) \approx_\ell \nu \text{pkaCh1}.(V_A\{^b/_v\} \mid V_B\{^a/_v\} \mid \text{processK})$$

First phase - Fujioka *et al.*

- On the left: $\nu pkaCh1.(V_A\{^a/_v\} \mid V_B\{^b/_v\} \mid processK)$

$$P \xrightarrow{in(skvaCh,skva)} P_1 \xrightarrow{in(skvbCh,skvb)} P_2 \rightarrow^*$$
$$\xrightarrow{\nu x_1.out(ch,x_1)} \nu b_A, r_A, b_B, r_B.(P_3 \mid \{^{(hostva,sign(blind(commit(a,r_A),b_A),skva))}/_{x_1}\})$$
$$\xrightarrow{\nu x_2.out(ch,x_2)} \nu b_A, r_A, b_B, r_B.(P_4 \mid \{^{(hostva,sign(blind(commit(a,r_A),b_A),skva))}/_{x_1}\} \mid \{^{(hostvb,sign(blind(commit(b,r_B),b_B),skvb)}/_{x_2}\})$$

- On the right: $\nu pkaCh1.(V_A\{^b/_v\} \mid V_B\{^a/_v\} \mid processK)$

$$Q \xrightarrow{in(skvaCh,skva)} Q_1 \xrightarrow{in(skvbCh,skvb)} Q_2 \rightarrow^*$$
$$\xrightarrow{\nu x_1.out(ch,x_1)} \nu b_A.\nu r_A.\nu b_B.\nu r_B.(Q_3 \mid \{^{(hostva,sign(blind(commit(b,r_A),b_A),skva))}/_{x_1}\})$$
$$\xrightarrow{\nu x_2.out(ch,x_2)} \nu b_A.\nu r_A.\nu b_B.\nu r_B.(Q_4 \mid \{^{(hostva,sign(blind(commit(b,r_A),b_A),skva))}/_{x_1}\} \mid \{^{(hostvb,sign(blind(commit(a,r_B),b_B),skvb)}/_{x_2}\})$$

$\longrightarrow$ $V_A\{^a/_v\}$ (on the left) has been imitated by $V_A\{^b/_v\}$ (on the right),
and $V_B\{^b/_v\}$ (on the left) has been imitated by $V_B\{^a/_v\}$ (on the right).

Second phase - Fujioka *et al.*

- On the left: $\nu pkaCh1.(V_A\{^a/_v\} \mid V_B\{^b/_v\} \mid processK)$

$$\phi_{P'} \equiv \nu b_A.\nu r_A.\nu b_B.\nu r_B. \quad \begin{array}{l} \{^{(hostva,sign(blind(commit(a,r_A),b_A),skva))}/_{x_1}\} \mid \\ \{^{(hostvb,sign(blind(commit(b,r_B),b_B),skvb))}/_{x_2}\} \mid \\ \{^{(commit(a,r_A),sign(commit(a,r_A),ska))}/_{x_3}\} \mid \\ \{^{(commit(b,r_B),sign(commit(b,r_B),ska))}/_{x_4}\} \end{array}$$

- On the right: $\nu pkaCh1.(V_A\{^b/_v\} \mid V_B\{^a/_v\} \mid processK)$

$$\phi_{Q'} \equiv \nu b_A.\nu r_A.\nu b_B.\nu r_B. \quad \begin{array}{l} \{^{(hostva,sign(blind(commit(b,r_A),b_A),skva))}/_{x_1}\} \mid \\ \{^{(hostvb,sign(blind(commit(a,r_B),b_B),skvb))}/_{x_2}\} \mid \\ \{^{(commit(a,r_B),sign(commit(a,r_B),ska))}/_{x_3}\} \mid \\ \{^{(commit(b,r_A),sign(commit(b,r_A),ska))}/_{x_4}\} \end{array}$$

$\longrightarrow$ $V_A\{^a/_v\}$ (on the left) has been imitated by $V_B\{^a/_v\}$ (on the right),
and $V_B\{^b/_v\}$ (on the left) has been imitated by $V_A\{^b/_v\}$ (on the right).

Third phase - Fujioka *et al.*

- On the left: $\nu pkaCh1.(V_A\{^a/_v\} \mid V_B\{^b/_v\} \mid processK)$

$$\phi_{P''} \equiv \nu b_A.\nu r_A.\nu b_B.\nu r_B. \begin{array}{l} \{^{(hostva, sign(blind(commit(a,r_A),b_A),skva))}/_{x_1}\} \\ \{^{(hostvb, sign(blind(commit(b,r_B),b_B),skvb))}/_{x_2}\} \mid \\ \{^{(commit(a,r_A), sign(commit(a,r_A),ska))}/_{x_3}\} \mid \\ \{^{(commit(b,r_B), sign(commit(b,r_B),ska))}/_{x_4}\} \mid \\ \{^{(l_A,r_A)}/_{x_5}\} \mid \{^{(l_B,r_B)}/_{x_6}\} \end{array}$$

- On the right: $\nu pkaCh1.(V_A\{^b/_v\} \mid V_B\{^a/_v\} \mid processK)$

$$\phi_{Q''} \equiv \nu b_A.\nu r_A.\nu b_B.\nu r_B. \begin{array}{l} \{^{(hostva, sign(blind(commit(b,r_A),b_A),skva))}/_{x_1}\} \mid \\ \{^{(hostvb, sign(blind(commit(a,r_B),b_B),skvb))}/_{x_2}\} \mid \\ \{^{(commit(a,r_B), sign(commit(a,r_B),ska))}/_{x_3}\} \mid \\ \{^{(commit(b,r_A), sign(commit(b,r_A),ska))}/_{x_4}\} \mid \\ \{^{(l_A,r_B)}/_{x_5}\} \mid \{^{(l_B,r_A)}/_{x_6}\} \end{array}$$

$\longrightarrow$ Again, voters voting in the same way simulated each other (as in the previous phase).

Receipt-freeness: Leaking secrets to the coercer

To model receipt-freeness we need to specify that a coerced voter cooperates with the coercer by leaking secrets on a channel ch

We denote by V^{ch} the process built from the process V as follows:

- $0^{ch} \mathrel{\widehat{=}} 0$,
- $(P \mid Q)^{ch} \mathrel{\widehat{=}} P^{ch} \mid Q^{ch}$,
- $(\nu n.P)^{ch} \mathrel{\widehat{=}} \nu n.\mathsf{out}(ch, n).P^{ch}$,
- $(\mathsf{in}(u, x).P)^{ch} \mathrel{\widehat{=}} \mathsf{in}(u, x).\mathsf{out}(ch, x).P^{ch}$,
- $(\mathsf{out}(u, M).P)^{ch} \mathrel{\widehat{=}} \mathsf{out}(u, M).P^{ch}$,
- ...

We denote by $V^{\backslash out(ch,\cdot)} \mathrel{\widehat{=}} \nu ch.(V \mid !\mathsf{in}(ch, x))$.

Receipt-freeness

Definition (Receipt-freeness)

A voting protocol is receipt-free if there exists a process V', satisfying

- $V'^{\backslash out(chc,\cdot)} \approx V_A\{^a/_v\}$,
- $S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}] \approx S[V' \mid V_B\{^c/_v\}]$.

Intuitively, there exists a process V' which

- does vote a,
- leaks (possibly fake) secrets to the coercer,
- and makes the coercer believe he voted c

Naive example 1

Voter process

$$V = \mathrm{out}(ch, \{v\}_{\mathrm{pub}(s)})$$

What about receipt-freenes?

i.e. Does there exists V' such that

- $V'^{\setminus out(chc,\cdot)} \approx V_A\{^a/_v\}$,
- $V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\} \approx V' \mid V_B\{^c/_v\}$.

The voter does not use any secret data (private key, nonce ...). Hence, the process $V' = V_A\{^a/_v\}$ satisfies the requirements.

- $V_A\{^a/_v\}^{\setminus out(chc,\cdot)} \approx V_A\{^a/_v\}$,
- $V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\} \approx V_A\{^a/_v\} \mid V_B\{^c/_v\}$.

$\longrightarrow$ OK

Naive example 1

Voter process

$$V = \mathrm{out}(ch, \{v\}_{\mathrm{pub}(s)})$$

What about receipt-freenes?

i.e. Does there exists V' such that

- $V'^{\backslash out(chc,\cdot)} \approx V_A\{^a/_v\}$,
- $V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\} \approx V' \mid V_B\{^c/_v\}$.

The voter does not use any secret data (private key, nonce ...). Hence, the process $V' = V_A\{^a/_v\}$ satisfies the requirements.

- $V_A\{^a/_v\}^{\backslash out(chc,\cdot)} \approx V_A\{^a/_v\}$,
- $V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\} \approx V_A\{^a/_v\} \mid V_B\{^c/_v\}$.

$\longrightarrow$ OK

Other examples

Naive example 2 (with probabilistic encryption)

Voter process

$$V(Id) = \nu r.\mathrm{out}(ch, \langle Id, \{v\}^{r}_{\mathrm{pub}(s)}\rangle)$$

What about receipt-freenes?

$\longrightarrow$ NOT OK since r can be used as a receipt

Protocol due to Fujioka *et al.*
What about receipt-freenes?

$\longrightarrow$ NOT OK since the blinding b_A and the commitment r_A can be used as a receipt

Other examples

Naive example 2 (with probabilistic encryption)

Voter process

$$V(Id) = \nu r.\text{out}(ch, \langle Id, \{v\}^{r}_{\text{pub}(s)}\rangle)$$

What about receipt-freenes?

$\longrightarrow$ NOT OK since r can be used as a receipt

Protocol due to Fujioka *et al.*
What about receipt-freenes?

$\longrightarrow$ NOT OK since the blinding b_A and the commitment r_A can be used as a receipt

Other examples

Naive example 2 (with probabilistic encryption)

Voter process

$$V(Id) = \nu r.\text{out}(ch, \langle Id, \{v\}^{r}_{\text{pub}(s)}\rangle)$$

What about receipt-freenes?

$\longrightarrow$ NOT OK since r can be used as a receipt

Protocol due to Fujioka *et al.*
What about receipt-freenes?

$\longrightarrow$ NOT OK since the blinding b_A and the commitment r_A can be used as a receipt

Summary

Coercion-Resistance is defined in a similar way (the voter has to used the outputs provided by the coercer)

Lemma

Let VP be a voting protocol. We have formally shown that: VP is coercion-resistant $\Longrightarrow$ VP is receipt-free $\Longrightarrow$ VP respects privacy.

Case Study (1): protocol due to Fujioka *et al.*

- We have established privacy
  $\hookrightarrow$ holds even if the authorities are corrupt
- This protocol is not receipt-free
  $\hookrightarrow$ the random numbers for blinding and commitment can be used as a receipt

Some additional case studies

Case Study (2): Protocol due to Okamoto

- We have established privacy and receipt-free
  $\hookrightarrow$ the random numbers for commitment can not be used as a receipt since there is a trapdoor.

$$\begin{aligned} \mathsf{open}(\mathsf{tdcommit}(m, r, td), r) &= m \\ \mathsf{tdcommit}(m_1, r, td) &= \mathsf{tdcommit}(m_2, f(m_1, r, td, m_2), td) \end{aligned}$$

- This protocol is not coercion-resistant
  $\hookrightarrow$ the commitment can be provided by the coercer without revealing the trapdoor to the voter.

Case Study (3): Protocol due to Lee *et al.*

- protocol based on re-encryption and designated verifier proofs,
- coercion-resistance holds

Outline of the talk

An existing tool (ProVerif)

Labeled bisimilarity ($\approx_\ell$)

The largest symmetric relation $\mathcal{R}$ on processes, such that $A \mathrel{\mathcal{R}} B$ implies

1. $\phi(A) \approx_s \phi(B)$ (depends on E),
2. if $A \rightarrow A'$, then $B \rightarrow^* B'$ and $A' \mathrel{\mathcal{R}} B'$ for some B',
3. if $A \xrightarrow{\alpha} A'$, then $B \rightarrow^* \xrightarrow{\alpha} \rightarrow^* B'$ and $A' \mathrel{\mathcal{R}} B'$ for some B'.

This relation is in general undecidable. Why?

- unfolding tree is infinite in depth
- unfolding tree is infinititely branching (because of inputs)
- equational theories may be complex

Tool: Proverif

$\longrightarrow$ Obviously, the procedure is not complete.

An existing tool (ProVerif)

Labeled bisimilarity ($\approx_\ell$)

The largest symmetric relation $\mathcal{R}$ on processes, such that $A \mathrel{\mathcal{R}} B$ implies

1. $\phi(A) \approx_s \phi(B)$ (depends on E),
2. if $A \rightarrow A'$, then $B \rightarrow^* B'$ and $A' \mathrel{\mathcal{R}} B'$ for some B',
3. if $A \xrightarrow{\alpha} A'$, then $B \rightarrow^* \xrightarrow{\alpha} \rightarrow^* B'$ and $A' \mathrel{\mathcal{R}} B'$ for some B'.

This relation is in general undecidable. Why?

- unfolding tree is infinite in depth
- unfolding tree is infinititely branching (because of inputs)
- equational theories may be complex

Tool: Proverif

$\longrightarrow$ Obviously, the procedure is not complete.

An existing tool (ProVerif)

Labeled bisimilarity ($\approx_\ell$)

The largest symmetric relation $\mathcal{R}$ on processes, such that $A \mathrel{\mathcal{R}} B$ implies

1. $\phi(A) \approx_s \phi(B)$ (depends on E),
2. if $A \to A'$, then $B \to^* B'$ and $A' \mathrel{\mathcal{R}} B'$ for some B',
3. if $A \xrightarrow{\alpha} A'$, then $B \to^* \xrightarrow{\alpha} \to^* B'$ and $A' \mathrel{\mathcal{R}} B'$ for some B'.

This relation is in general undecidable. Why?

- unfolding tree is infinite in depth
- unfolding tree is infinititely branching (because of inputs)
- equational theories may be complex

Tool: Proverif

$\longrightarrow$ Obviously, the procedure is not complete.

Drawbacks of ProVerif

Proverif is not able to establish privacy for the naive vote protocol

$$\{a\}_{\mathrm{pub}(S)} \mid \{b\}_{\mathrm{pub}(S)} \approx \{b\}_{\mathrm{pub}(S)} \mid \{a\}_{\mathrm{pub}(S)}$$

... and more generally for any electronic voting protocols.

Why?

- ProVerif works on biprocesses (processes having the same structure).

$P \approx Q \quad \Leftrightarrow \quad$ let bool = choice[true,false] in
if bool = true then P else Q

- Technique relies on easily matching up the execution paths of the two processes

First Phase $\quad V_A\{^a/_v\} \mid V_B\{^b/_v\} \approx V_A\{^b/_v\} \mid V_B\{^a/_v\}$

Second Phase $\quad V_A\{^a/_v\} \mid V_B\{^b/_v\} \approx V_A\{^b/_v\} \mid V_B\{^a/_v\}$

Drawbacks of ProVerif

Proverif is not able to establish privacy for the naive vote protocol

$$\{a\}_{\mathrm{pub}(S)} \mid \{b\}_{\mathrm{pub}(S)} \approx \{b\}_{\mathrm{pub}(S)} \mid \{a\}_{\mathrm{pub}(S)}$$

... and more generally for any electronic voting protocols.

Why?

- ProVerif works on biprocesses (processes having the same structure).

$$P \approx Q \quad \Leftrightarrow \quad \begin{array}{l} \text{let bool} = \text{choice[true,false] in} \\ \text{if bool} = \text{true then P else Q} \end{array}$$

- Technique relies on easily matching up the execution paths of the two processes

First Phase $\quad V_A\{^a/_v\} \mid V_B\{^b/_v\} \approx V_A\{^b/_v\} \mid V_B\{^a/_v\}$

Second Phase $\quad V_A\{^a/_v\} \mid V_B\{^b/_v\} \approx V_A\{^b/_v\} \mid V_B\{^a/_v\}$

Drawbacks of ProVerif

Proverif is not able to establish privacy for the naive vote protocol

$$\{a\}_{\mathrm{pub}(S)} \mid \{b\}_{\mathrm{pub}(S)} \approx \{b\}_{\mathrm{pub}(S)} \mid \{a\}_{\mathrm{pub}(S)}$$

... and more generally for any electronic voting protocols.

Why?

- ProVerif works on biprocesses (processes having the same structure).

$P \approx Q \quad \Leftrightarrow \quad$ let bool = choice[true,false] in
if bool = true then P else Q

- Technique relies on easily matching up the execution paths of the two processes

First Phase $\quad V_A\{^a/_v\} \mid V_B\{^b/_v\} \approx V_A\{^b/_v\} \mid V_B\{^a/_v\}$

Second Phase $\quad V_A\{^a/_v\} \mid V_B\{^b/_v\} \approx V_A\{^b/_v\} \mid V_B\{^a/_v\}$

Drawbacks of ProVerif

Proverif is not able to establish privacy for the naive vote protocol

$$\{a\}_{\mathrm{pub}(S)} \mid \{b\}_{\mathrm{pub}(S)} \approx \{b\}_{\mathrm{pub}(S)} \mid \{a\}_{\mathrm{pub}(S)}$$

... and more generally for any electronic voting protocols.

Why?

- ProVerif works on biprocesses (processes having the same structure).

  $P \approx Q \quad \Leftrightarrow \quad$ let bool = choice[true,false] in
if bool = true then P else Q

- Technique relies on easily matching up the execution paths of the two processes

  First Phase $\quad V_A\{^a/_v\} \mid V_B\{^b/_v\} \approx V_A\{^b/_v\} \mid V_B\{^a/_v\}$

  Second Phase $\quad V_A\{^a/_v\} \mid V_B\{^b/_v\} \approx V_A\{^b/_v\} \mid V_B\{^a/_v\}$

First approach: procedure based on ProVerif

$\longrightarrow$ with Mark Ryan and Ben Smith (University of Birmingham)

$$V_A\{^a/_v\} \mid V_B\{^b/_v\} \approx V_A\{^b/_v\} \mid V_B\{^a/_v\}$$

where $V_X = V_X^1; phase1; V_X^2$

Skip Details

First approach: procedure based on ProVerif

⟶ with Mark Ryan and Ben Smith (University of Birmingham)

$$V_A\{^a/_v\} \mid V_B\{^b/_v\} \approx V_A\{^b/_v\} \mid V_B\{^a/_v\}$$

where $V_X = V_X^1; phase1; V_X^2$

Conjecture

To establish the equivalence, it may be sufficient to show that

- $V_A^1\{^a/_v\} \mid V_B^1\{^b/_v\} \approx V_A^1\{^b/_v\} \mid V_B^1\{^a/_v\}$, (1st phase)
- for all interleaving l_1 of $V_A^1\{^a/_v\} \mid V_B^1\{^b/_v\}$, there exists an interleaving l_2 of $V_A^1\{^b/_v\} \mid V_B^1\{^a/_v\}$ such that (2nd phase)

  $$l_1; phase1; (V_A^2\{^a/_v\} \mid V_B^2\{^b/_v\}) \approx l_2; phase1; (V_B^2\{^a/_v\} \mid V_A^2\{^b/_v\})$$

  and vice-versa,
- and some additional assumptions.

Second approach: symbolic bisimulation

⟶ with Steve Kremer (LSV) and Mark Ryan (University of Birmingham)

Our Goal:
to do better than Proverif in the context of a bounded number of sessions

- Infinite depth:
  ↪ we restrict to consider processes without replication.
- Infinite branching:
  ↪ define a notion of symbolic processes and symbolic bisimulation

▸ Skip Details

Symbolic Bisimulation

Concrete Side:

$$\nu s, k.(\mathrm{in}(c,x); P \mid \{^{\{s\}_k}/_y\}) \xrightarrow{in(c,m_1)} \nu s, k.(P\{^{m_1}/_x\} \mid \{^{\{s\}_k}/_y\})$$

Symbolic Side:

$$(\nu s, k.(\mathrm{in}(c,x); P \mid \{^{\{s\}_k}/_y\}) \; ; \; \mathcal{C}) \xrightarrow{in(c,x)}$$
$$(\nu s, k.(P \mid \{^{\{s\}_k}/_y\}) \; ; \; \mathcal{C} \cup \{\nu s, k.\{^{\{s\}_k}/_y\} \Vdash x\})$$

Definition

Symbolic bisimulation $\approx_{symb}$ is the largest symmetric relation $\mathcal{R}$ such that $(A \; ; \; \mathcal{C}_A) \; \mathcal{R} \; (B \; ; \; \mathcal{C}_B)$ implies

- $\mathcal{C}_A$ and $\mathcal{C}_B$ are E-equivalent,
- if $(A \; ; \; \mathcal{C}_A) \rightarrow_s (A' \; ; \; \mathcal{C}'_A)$ with $Sol_{\mathsf{E}}(\mathcal{C}'_A) \neq \emptyset$ then there exists $(B' \; ; \; \mathcal{C}'_B)$ such that $(B \; ; \; \mathcal{C}_B) \rightarrow_s^* (B' \; ; \; \mathcal{C}'_B)$ and $(A' \; ; \; \mathcal{C}'_A) \; \mathcal{R} \; (B' \; ; \; \mathcal{C}'_B)$
- if $(A \; ; \; \mathcal{C}_A) \xrightarrow{\alpha}_s (A' \; ; \; \mathcal{C}'_A)$ ...

Symbolic Bisimulation

Concrete Side:

$$\nu s, k.(\text{in}(c,x); P \mid \{^{\{s\}_k}/_y\}) \xrightarrow{in(c,m_1)} \nu s, k.(P\{^{m_1}/_x\} \mid \{^{\{s\}_k}/_y\})$$

Symbolic Side:

$$(\nu s, k.(\text{in}(c,x); P \mid \{^{\{s\}_k}/_y\}) \; ; \; \mathcal{C}) \xrightarrow{in(c,x)}$$
$$(\nu s, k.(P \mid \{^{\{s\}_k}/_y\}) \; ; \; \mathcal{C} \cup \{\nu s, k.\{^{\{s\}_k}/_y\} \Vdash x\})$$

Definition

Symbolic bisimulation $\approx_{symb}$ is the largest symmetric relation $\mathcal{R}$ such that $(A \; ; \; \mathcal{C}_A) \; \mathcal{R} \; (B \; ; \; \mathcal{C}_B)$ implies

- $\mathcal{C}_A$ and $\mathcal{C}_B$ are E-equivalent,
- if $(A \; ; \; \mathcal{C}_A) \rightarrow_s (A' \; ; \; \mathcal{C}'_A)$ with $Sol_{\mathsf{E}}(\mathcal{C}'_A) \neq \emptyset$ then there exists $(B' \; ; \; \mathcal{C}'_B)$ such that $(B \; ; \; \mathcal{C}_B) \rightarrow^*_s (B' \; ; \; \mathcal{C}'_B)$ and $(A' \; ; \; \mathcal{C}'_A) \; \mathcal{R} \; (B' \; ; \; \mathcal{C}'_B)$
- if $(A \; ; \; \mathcal{C}_A) \xrightarrow{\alpha}_s (A' \; ; \; \mathcal{C}'_A)$ ...

Symbolic Bisimulation

Concrete Side:

$$\nu s, k.(\mathrm{in}(c,x); P \mid \{^{\{s\}_k}/_y\}) \xrightarrow{in(c,m_1)} \nu s, k.(P\{^{m_1}/_x\} \mid \{^{\{s\}_k}/_y\})$$

Symbolic Side:

$$(\nu s, k.(\mathrm{in}(c,x); P \mid \{^{\{s\}_k}/_y\}) \; ; \; \mathcal{C}) \xrightarrow{in(c,x)}$$
$$(\nu s, k.(P \mid \{^{\{s\}_k}/_y\}) \; ; \; \mathcal{C} \cup \{\nu s, k.\{^{\{s\}_k}/_y\} \Vdash x\})$$

Definition

Symbolic bisimulation $\approx_{symb}$ is the largest symmetric relation $\mathcal{R}$ such that $(A \; ; \; \mathcal{C}_A) \; \mathcal{R} \; (B \; ; \; \mathcal{C}_B)$ implies

- $\mathcal{C}_A$ and $\mathcal{C}_B$ are E-equivalent,
- if $(A \; ; \; \mathcal{C}_A) \rightarrow_s (A' \; ; \; \mathcal{C}'_A)$ with $Sol_{\mathsf{E}}(\mathcal{C}'_A) \neq \emptyset$ then there exists $(B' \; ; \; \mathcal{C}'_B)$ such that $(B \; ; \; \mathcal{C}_B) \rightarrow_s^* (B' \; ; \; \mathcal{C}'_B)$ and $(A' \; ; \; \mathcal{C}'_A) \; \mathcal{R} \; (B' \; ; \; \mathcal{C}'_B)$
- if $(A \; ; \; \mathcal{C}_A) \xrightarrow{\alpha}_s (A' \; ; \; \mathcal{C}'_A)$ ...

Main Result

Theorem

Let A and B be two processes. We have that

$$(A \; ; \; \emptyset) \approx_{symb} (B \; ; \; \emptyset) \implies A \approx_{\ell} B$$

Sources of Incompleteness

$\hookrightarrow$ due to the fact that the instanciation of an input variable is postponed until the moment it is actually used

Example: $P_1 \approx_\ell Q_1$ whereas $(P_1 \; ; \; \emptyset) \not\approx_{symb} (Q_1 \; ; \; \emptyset)$.

$$P_1 = \nu c_1.\mathrm{in}(c_2, x).(\mathrm{out}(c_1, b) \mid \mathrm{in}(c_1, y) \mid \text{if } x = a \text{ then } \mathrm{in}(c_1, z).\mathrm{out}(c_2, a))$$
$$Q_1 = \nu c_1.\mathrm{in}(c_2, x).(\mathrm{out}(c_1, b) \mid \mathrm{in}(c_1, y) \mid \mathrm{in}(c_1, z).\text{if } x = a \text{ then } \mathrm{out}(c_2, a))$$

$\hookrightarrow$ but we think that our symbolic bisimulation is complete enough to deal with many interesting cases.

Main Result

Theorem

Let A and B be two processes. We have that

$$(A\ ;\ \emptyset) \approx_{symb} (B\ ;\ \emptyset) \implies A \approx_\ell B$$

Sources of Incompleteness

$\hookrightarrow$ due to the fact that the instanciation of an input variable is postponed until the moment it is actually used

Example: $P_1 \approx_\ell Q_1$ whereas $(P_1\ ;\ \emptyset) \not\approx_{symb} (Q_1\ ;\ \emptyset)$.

$$P_1 = \nu c_1.\mathrm{in}(c_2, x).(\mathrm{out}(c_1, b) \mid \mathrm{in}(c_1, y) \mid \text{if } x = a \text{ then } \mathrm{in}(c_1, z).\mathrm{out}(c_2, a))$$
$$Q_1 = \nu c_1.\mathrm{in}(c_2, x).(\mathrm{out}(c_1, b) \mid \mathrm{in}(c_1, y) \mid \mathrm{in}(c_1, z).\text{if } x = a \text{ then } \mathrm{out}(c_2, a))$$

$\hookrightarrow$ but we think that our symbolic bisimulation is complete enough to deal with many interesting cases.

Main Result

Theorem

Let A and B be two processes. We have that

$$(A\ ;\ \emptyset) \approx_{symb} (B\ ;\ \emptyset) \implies A \approx_\ell B$$

Sources of Incompleteness

↪ due to the fact that the instanciation of an input variable is postponed until the moment it is actually used

Example: $P_1 \approx_\ell Q_1$ whereas $(P_1\ ;\ \emptyset) \not\approx_{symb} (Q_1\ ;\ \emptyset)$.

$$P_1 = \nu c_1.\mathrm{in}(c_2, x).(\mathrm{out}(c_1, b) \mid \mathrm{in}(c_1, y) \mid \text{if } x = a \text{ then } \mathrm{in}(c_1, z).\mathrm{out}(c_2, a))$$
$$Q_1 = \nu c_1.\mathrm{in}(c_2, x).(\mathrm{out}(c_1, b) \mid \mathrm{in}(c_1, y) \mid \mathrm{in}(c_1, z).\text{if } x = a \text{ then } \mathrm{out}(c_2, a))$$

↪ but we think that our symbolic bisimulation is complete enough to deal with many interesting cases.

Conclusion and Future Works

Conclusion:

- First formal definitions of receipt-freeness and coercion-resistance
- 3 case studies giving interesting insights
- A notion of symbolic bisimulation that is sound w.r.t. the concrete one

Works in Progress:

- An automatic procedure based on ProVerif

Future Works:

- to design a procedure to solve our constaint systems for a class of equational theory as larger as possible
- to implement a tool based on this approach,
- individual/universal verifiability

Conclusion and Future Works

Conclusion:

- First formal definitions of receipt-freeness and coercion-resistance
- 3 case studies giving interesting insights
- A notion of symbolic bisimulation that is sound w.r.t. the concrete one

Works in Progress:

- An automatic procedure based on ProVerif

Future Works:

- to design a procedure to solve our constaint systems for a class of equational theory as larger as possible
- to implement a tool based on this approach,
- individual/universal verifiability