Modelling and verifying privacy-type properties of electroning voting protocols

Stéphanie Delaune

Post-doctorante au LORIA – Projet Cassis

Monday 12th March

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)

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

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

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

Voting phase:

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

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

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

Voting phase:

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

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

Security properties ...

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 $\Rightarrow$ 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)

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

Motivation for using the applied π-calculus

Applied pi-calculus: [Abadi & Fournet, 01]
basic programming language with constructs for concurrency and communication

- based on the π-calculus [Milner *et al.*, 92]
- in some ways similar to the spi-calculus [Abadi & Gordon, 98]

Advantages:

- allows us to model less classical cryptographic primitives
- both reachability and equivalence-based specification of properties
- automated proofs using ProVerif tool [Blanchet]
- powerful proof techniques for hand proofs
- successfully used to analyze a variety of security protocols

Motivation for using the applied π-calculus

Applied pi-calculus: [Abadi & Fournet, 01]

basic programming language with constructs for concurrency and communication

- based on the π-calculus [Milner *et al.*, 92]
- in some ways similar to the spi-calculus [Abadi & Gordon, 98]

Advantages:

- allows us to model less classical cryptographic primitives
- both reachability and equivalence-based specification of properties
- automated proofs using ProVerif tool [Blanchet]
- powerful proof techniques for hand proofs
- successfully used to analyze a variety of security protocols

The applied π-calculus on an example

Syntax:

- Equational theory: $dec(enc(x, y), y) = x$
- Process:

$$P = \nu s, k.(\text{out}(c_1, enc(s, k)) \mid \text{in}(c_1, y).\text{out}(c_2, dec(y, k))).$$

Semantics:

- Operational semantics $\rightarrow$: closed by structural equivalence ($\equiv$) and application of evaluation contexts such that

Comm	$\text{out}(a, x).P \mid \text{in}(a, x).Q \rightarrow P \mid Q$
Then	if $M = M$ then P else $Q \rightarrow P$
Else	if $M = N$ then P else $Q \rightarrow Q \quad (M \neq_{\mathsf{E}} N)$

Example: $P \rightarrow \nu s, k.\text{out}(c_2, s)$

- Labeled operational semantics $\xrightarrow{\alpha}$

The applied π-calculus on an example

Syntax:

- Equational theory: $dec(enc(x,y),y) = x$
- Process:

$$P = \nu s, k.(\text{out}(c_1, enc(s,k)) \mid \text{in}(c_1, y).\text{out}(c_2, dec(y,k))).$$

Semantics:

- Operational semantics $\rightarrow$: closed by structural equivalence ($\equiv$) and application of evaluation contexts such that

Comm	$\text{out}(a,x).P \mid \text{in}(a,x).Q \rightarrow P \mid Q$
Then	$\text{if } M = M \text{ then } P \text{ else } Q \rightarrow P$
Else	$\text{if } M = N \text{ then } P \text{ else } Q \rightarrow Q \quad (M \neq_{\mathsf{E}} N)$

Example: $P \rightarrow \nu s, k.\text{out}(c_2, s)$

- Labeled operational semantics $\xrightarrow{\alpha}$

The applied π-calculus on an example

Syntax:

- Equational theory: $dec(enc(x, y), y) = x$
- Process:

$$P = \nu s, k.(\text{out}(c_1, enc(s, k)) \mid \text{in}(c_1, y).\text{out}(c_2, dec(y, k))).$$

Semantics:

- Operational semantics $\rightarrow$: closed by structural equivalence ($\equiv$) and application of evaluation contexts such that

Comm	$\text{out}(a, x).P \mid \text{in}(a, x).Q \rightarrow P \mid Q$
Then	$\text{if } M = M \text{ then } P \text{ else } Q \rightarrow P$
Else	$\text{if } M = N \text{ then } P \text{ else } Q \rightarrow Q \quad (M \neq_{\mathsf{E}} N)$

Example: $P \rightarrow \nu s, k.\text{out}(c_2, s)$

- Labeled operational semantics $\xrightarrow{\alpha}$

The applied π-calculus on an example

Syntax:

- Equational theory: $dec(enc(x, y), y) = x$
- Process:

$$P = \nu s, k.(\text{out}(c_1, enc(s, k)) \mid \text{in}(c_1, y).\text{out}(c_2, dec(y, k))).$$

Semantics:

- Operational semantics $\rightarrow$: closed by structural equivalence ($\equiv$) and application of evaluation contexts such that

Comm	$\text{out}(a, x).P \mid \text{in}(a, x).Q \rightarrow P \mid Q$
Then	$\text{if } M = M \text{ then } P \text{ else } Q \rightarrow P$
Else	$\text{if } M = N \text{ then } P \text{ else } Q \rightarrow Q \quad (M \neq_{\mathsf{E}} N)$

Example: $P \;\rightarrow\; \nu s, k.\text{out}(c_2, s)$

- Labeled operational semantics $\xrightarrow{\alpha}$

Equivalences on processes

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 $C[\,]$.

Labeled bisimilarity ($\approx_\ell$)

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

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

Equivalences on processes

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 $C[\,]$.

Labeled bisimilarity ($\approx_\ell$)

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

1. $\phi(A) \approx_s \phi(B)$ (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'.

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)).
```

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

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

Some Examples

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

Naive vote protocol (version 1)

$$V \rightarrow S : \{v\}_{\mathrm{pub}(S)}$$

What about privacy?

Some Examples

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

Naive vote protocol (version 1)

$$V \rightarrow S : \{v\}_{\mathsf{pub}(S)}$$

What about privacy? OK

Some Examples

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

Naive vote protocol (version 1)

$$V \rightarrow S : \{v\}_{\mathsf{pub}(S)}$$

What about privacy? OK

Naive vote protocol (version 2)

$$V \rightarrow S : Id, \{v\}_{\mathsf{pub}(S)}$$

What about privacy?

Some Examples

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

Naive vote protocol (version 1)

$$V \rightarrow S : \{v\}_{\mathsf{pub}(S)}$$

What about privacy? OK

Naive vote protocol (version 2)

$$V \rightarrow S : Id, \{v\}_{\mathsf{pub}(S)}$$

What about privacy?

- deterministic encryption: NOT OK
- probabilistic encryption: OK

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

First Phase:

the voter gets a "token" from the administrator.

$$\begin{array}{llll} 1. & V \rightarrow A & : & V, sign(blind(commit(vote, r), b), V) \\ 2. & A \rightarrow V & : & sign(blind(commit(vote, r), b), A) \end{array}$$

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

Voting phase:

$$\begin{array}{llll} 3. & V \rightarrow C & : & sign(commit(vote, r), A) \\ 4. & C \rightarrow & : & l, sign(commit(vote, r), A) \end{array}$$

Counting phase:

$$\begin{array}{llll} 5. & V \rightarrow C & : & l, r \end{array}$$

6. C publishes the outcome of the vote

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

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

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): 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

Outline of the talk

An existing tool (ProVerif)

Labeled bisimilarity ($\approx_\ell$)

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

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

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

This relation is in general undecidable. Why?

- unfolding tree is infinite in depth
- unfolding tree is infinitite{ly} 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 $A \mathcal{R} B$ implies

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

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

... 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).

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

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

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

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

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

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

$\longrightarrow$ 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 $V_X = V_X^1; phase1; V_X^2$

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

$\longrightarrow$ 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:
  $\hookrightarrow$ we restrict to consider processes without replication.
- Infinite branching:
  $\hookrightarrow$ 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:

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

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:

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

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

Conjecture

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.

Main Result

Conjecture

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

Conjecture

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 $(P_1 \; ; \; \emptyset) \not\approx_{symb} (Q_1 \; ; \; \emptyset)$.

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

$\hookrightarrow$ 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
- Coercion-Resistance $\Rightarrow$ Receipt-Freeness $\Rightarrow$ Privacy,
- 3 Case studies giving interesting insights

Works in Progress:

- An automatic procedure based on ProVerif
- A symbolic bisimulation for the applied pi calculus

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,
- other properties based on *not being able to prove* (abuse freeness)

Conclusion and Future Works

Conclusion:

- First formal definitions of receipt-freeness and coercion-resistance
- Coercion-Resistance $\Rightarrow$ Receipt-Freeness $\Rightarrow$ Privacy,
- 3 Case studies giving interesting insights

Works in Progress:

- An automatic procedure based on ProVerif
- A symbolic bisimulation for the applied pi calculus

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,
- other properties based on *not being able to prove* (abuse freeness)