Verifying Electronic Voting Protocols in the Applied Pi Calculus

Mark Ryan
University of Birmingham

based on joint work with

Stéphanie Delaune
Steve Kremer
Mounira Kourjieh
Ben Smyth

VETO'09

Outline

How could it be secure?

Two approaches

Trusted client crypto

- Blind signatures

  Fujioka/Okamoto/Ohta 92,
  Okamoto 96, 97
  Ohkubu/Miura/Abe/Fujioka/Okamoto 99
  Canard/Gaud/Traoré 06

- Homomorphic encryption

  ElGamal
  Cohen/Fisher 85, Benaloh/Yung 86,
  Benaloh/Tuinstra 94, Sako/Kilian 94
  Cramer/Schoenmakers/Franklin/-
  Gennaro/Yung 96, 97
  Paillier
  Baudron/Fouque/Pointch./Stern/Poup.01,
  Damgard/Jurik 01

- Zero-knowledge proofs

  Juels/Catalano/Jakobsson 05

No client crypto

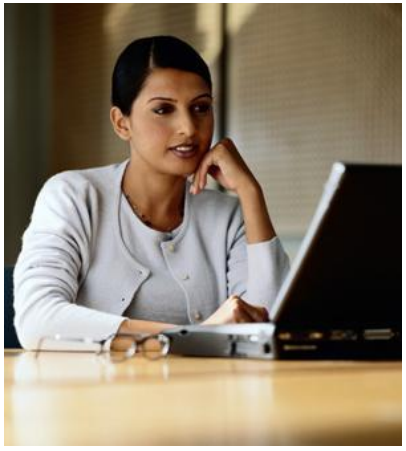
- Paper-and-scan
  + challenges

  P.Ryan, Schneider 05, 06, 08 *Prêt-à-Voter*
  Chaum, Clark, Popoveniuc 06 *Punchscan*

- touch-screen
  + challenges

  Neff 04,

Security by trusted client software



$\longrightarrow \longrightarrow \longrightarrow \longrightarrow \longrightarrow \longrightarrow \longrightarrow \longrightarrow \longrightarrow$

- trusted by user
- does not need to be trusted by authorities or other voters

- not trusted by user
- doesn't need to be trusted by anyone

Voting system: desired properties

- Eligibility: only legitimate voters can vote, and at most once (This also implies that the voting authorities cannot insert votes)
- Fairness: no early results can be obtained
- Privacy: the fact that a particular voter in a particular way is not revealed to anyone
- Receipt-freeness: a voter cannot later prove to a coercer that she voted in a certain way
- Coercion-resistance: a voter cannot interactively cooperate with a coercer to prove that she voted in a certain way
- Individual verifiability: a voter can verify that her vote was really counted
- Universal verifiability: a voter can verify that the published outcome really is the sum of all the votes

. . . and all this even in the presence of corrupt election authorities!

Attacker model

Ideally, we want to model a very powerful attacker:

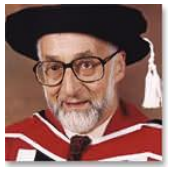
- It has "Dolev-Yao" capabilities, i.e.
  - it completely controls the communication channels, so it is able to record, alter, delete, insert, redirect, reorder, and reuse past or current messages, and inject new messages (The *network* is the attacker)
  - manipulate data in arbitrary ways, including applying crypto operations provided has the necessary keys
- It includes the election authorities.
- It includes the other voters.

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], but more general w.r.t. cryptography




Advantages:

- naturally models a Dolev-Yao attacker
- 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

Equations to model the cryptography

Examples

1. *Encryption and signatures*

$$\texttt{decrypt(}\ \texttt{encrypt(m,pk(k))}\texttt{,}\ \texttt{k}\ \texttt{)} \quad = \quad \texttt{m}$$
$$\texttt{checksign(}\ \texttt{sign(m,k)}\texttt{,}\ \texttt{m}\texttt{,}\ \texttt{pk(k)}\ \texttt{)} \quad = \quad \texttt{ok}$$

2. *Blind signatures*

$$\texttt{unblind(}\ \texttt{sign(}\ \texttt{blind(m,r)}\texttt{,}\ \texttt{sk}\ \texttt{)}\texttt{,}\ \texttt{r}\ \texttt{)} \quad = \quad \texttt{sign(m,sk)}$$

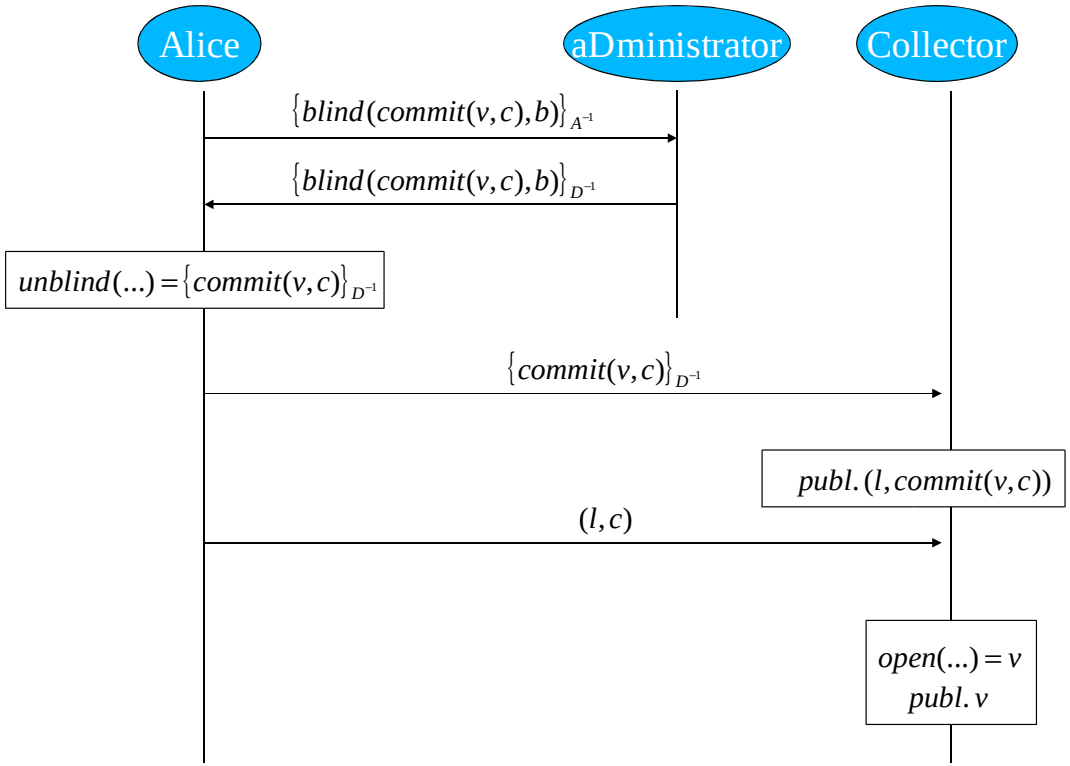
3. *Designated verifier proof of re-encryption*
   The term dvp(x,rencrypt(x,r),r,pkv) represents a proof designated for the owner of pkv that x and rencrypt(x,r) have the same plaintext.

$$\texttt{checkdvp(dvp(x,rencrypt(x,r),r,pkv),x,rencrypt(x,r),pkv)} = \texttt{ok}$$
$$\texttt{checkdvp(}\ \texttt{dvp(x,y,z,skv)}\texttt{,}\ \texttt{x}\texttt{,}\ \texttt{y}\texttt{,}\ \texttt{pk(skv)}\ \texttt{)} = \texttt{ok}.$$

Coding protocols as processes

Example ([FOO'92]):

```
processV =
  new b; new c;
  let bcv = blind(commit(v,c),b) in
  out(ch, (sign(bcv, skv)));
  in(ch,m2);
  if getMess(m2,pka)=bcv then
  let scv = unblind(m2,b) in
  str_phase 1;
  out(ch, scv);
  in(ch,(l, =scv));
  str_phase 2;
  out(ch,(l,c)).
```



Formalisation of vote-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

$\hookrightarrow$ consider two honest voters and swap their votes

Definition (Privacy)

A voting protocol respects privacy if

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

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

$P ::=$
0
$P \mid P$
$\nu n.P$
$\mathsf{in}(u, x).P$
$\mathsf{out}(u, M).P$
$\mathsf{if}\ M = N\ \mathsf{then}\ P\ \mathsf{else}\ P$
$!P$
$\ldots$

P^{ch} in terms of P

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

We denote by $P^{\backslash \mathsf{out}(chc,\cdot)}$ the process $\nu chc.(P \mid !\mathsf{in}(chc, x))$.

Lemma: $(P^{ch})^{\backslash \mathsf{out}(chc,\cdot)} \approx_\ell P$

Receipt-freeness: definition

Intuition

There exists a process V' which

- votes a,
- leaks (possibly fake) secrets to the coercer,
- and makes the coercer believe she voted c

Definition (Receipt-freeness)

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

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

Case study: Lee *et al.* protocol
We prove receipt-freeness by

- exhibiting V'
- showing that $V'\backslash out(chc,\cdot) \approx_\ell V_A\{^a/_v\}$
- showing that $S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}] \approx_\ell S[V' \mid V_B\{^c/_v\}]$

Coercion resistance: talking with the coercer

Like receipt-freness, but: voter interacts with the coercer during the protocol (instead of just supplying data at the end).

- The *voting booth* makes coercion resistance possible.

Interactively communicating with the coercer:

P^{c_1,c_2} in terms of P

- $0^{c_1,c_2} = 0$,
- $(P \mid Q)^{c_1,c_2} = P^{c_1,c_2} \mid Q^{c_1,c_2}$
- $(\nu n.P)^{c_1,c_2} = \nu n.\mathsf{out}(c_1, n).P^{c_1,c_2}$
- $(\mathsf{in}(u, x).P)^{c_1,c_2} = \mathsf{in}(u, x).\mathsf{out}(c_1, x).P^{c_1,c_2}$
- $(\mathsf{out}(u, M).P)^{c_1,c_2} = \mathsf{in}(c_2, x).\mathsf{out}(u, x).P^{c_1,c_2}$
- $(!P)^{c_1,c_2} = !P^{c_1,c_2}$,
- $(\text{if } M = N \text{ then } P \text{ else } Q)^{c_1,c_2} = \mathsf{in}(c_2, x).\ \text{if } x = \text{true then } P^{c_1,c_2} \text{ else } Q^{c_1,c_2}$

Coercion resistance: definition

Definition (Coercion resistance)

VP is coercion resistant if there exists a process V' such that for any $C = \nu c_1.\nu c_2.(\_ \mid P)$ satisfying

- $\tilde{n} \cap fn(C) = \emptyset$
- $S[C[V_A\{^?/_v\}^{c1,c2}] \mid V_B\{^a/_v\}] \approx_\ell S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}]$

we have

- $C[V']^{\backslash out(chc,\cdot)} \approx_\ell V_A\{^a/_v\}$,
- $S[C[V_A\{^?/_v\}^{c_1,c_2}] \mid V_B\{^a/_v\}] \approx_\ell S[C[V'] \mid V_B\{^c/_v\}]$.

Intuitively, C together with the environment represent the coercer. The definition says there's a strategy V' for the voter such that

- if the coercer is trying to force A to vote c

then

- A can do V', which will result in an a vote, but will satisfy the coercer.

Doesn't take account of *fault attacks* (cf. Küsters/Truderung).

Privacy properties

Proposition

Let VP be a voting protocol. Then

$$VP \text{ is coercion-resistant}$$
$$\Downarrow$$
$$VP \text{ is receipt-free}$$
$$\Downarrow$$
$$VP \text{ respects privacy}$$


Where is my
VOTE?
GETTY IMAGES

Election verifiability

Individual verifiability

A voter can check her own vote is included in the tally.

Universal verifiability

Anyone can check that the declared outcome corresponds to the tally.

Eligibility verifiability

Anyone can check that only eligible votes are included in the declared outcome.

Remarks

- Verifiability $\neq$ correctness
- What system components need to be trusted in order to carry out these checks?

Individual verifiability

Please check with me for an update on this part of the presentation!

Trustworthiness requirements

Property	*FOO'92*	*Oka. et al.'97*	*Lee et al.'03*
Vote-privacy trusted authorities	✓ none	✓ timelin. mbr.	✓ administrator
Receipt-freeness trusted authorities	× n/a	✓ timelin. mbr.	✓ admin. & collector
Coercion-resistance trusted authorities	× n/a	× n/a	✓ admin. & collector

	Trustworthiness requirement		
property	my client	other clients	server
privacy	yes	partly	no
coercion resistance	yes	partly	partly
individual verifiability	partly	no	no
universal verifiability	no	no	no
eligibility verifiability	no	no	no

Conclusions and future work

Conclusions

- First formal definitions of receipt-freeness and coercion-resistance
- coercion-resistance $\Rightarrow$ receipt-freeness $\Rightarrow$ privacy
- First *generic* formal definitions of election verifiability. Suitable for automation.

Future work

- Decision procedure for observational equivalence for processes without replication.
- Voting systems that are not client-crypto-based.