ProVerif/applied pi calculus analysis of electronic voting systems

Mark D. Ryan

based on joint work with
Stéphanie Delaune
Steve Kremer
Ben Smyth

FOSAD, August 2012

Outline

Electronic voting: potential

Electronic voting potentially offers

- Efficiency
  - higher voter participation
  - greater accuracy
  - lower costs
- Better security
  - vote-privacy even in presence of corrupt election authorities
  - voter verification, i.e. the ability of voters and observers to check the declared outcome against the votes cast.

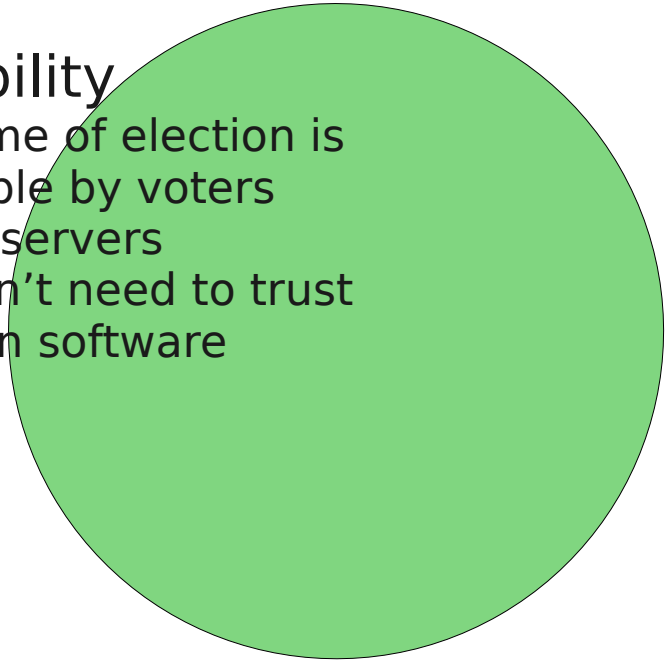
Governments world over have been trialling e-voting, e.g. USA, UK, Canada, Brasil, the Netherlands and Estonia.

Can also be useful for smaller-scale elections (student guild, shareholder voting, trade union ballots, local government).

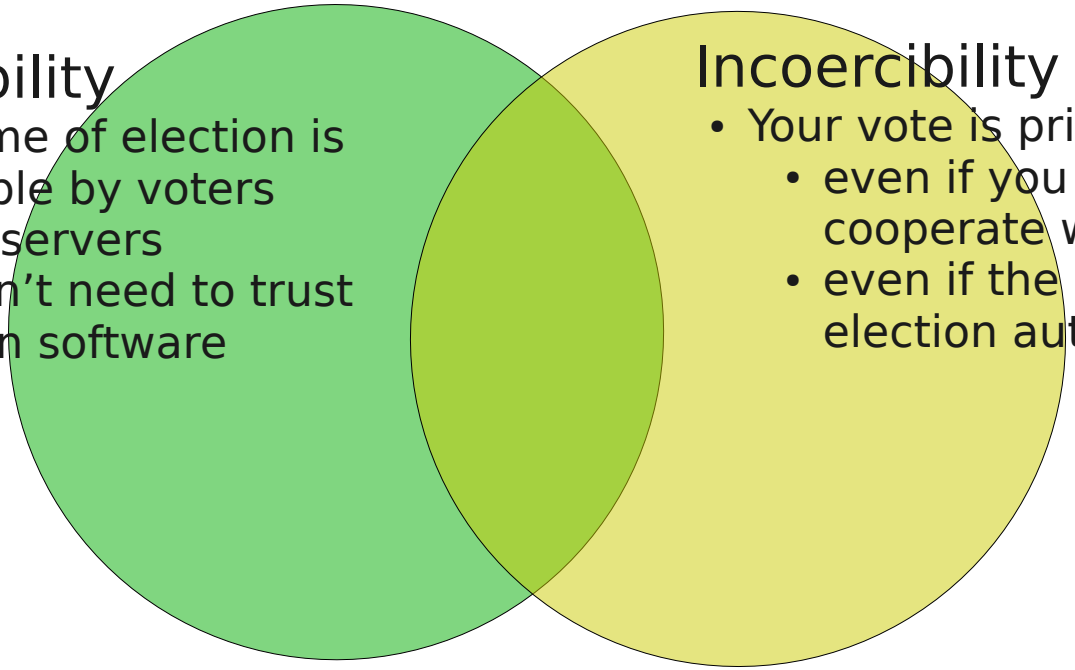
Desired properties

Verifiability

- Outcome of election is verifiable by voters and observers
- You don’t need to trust election software



Desired properties



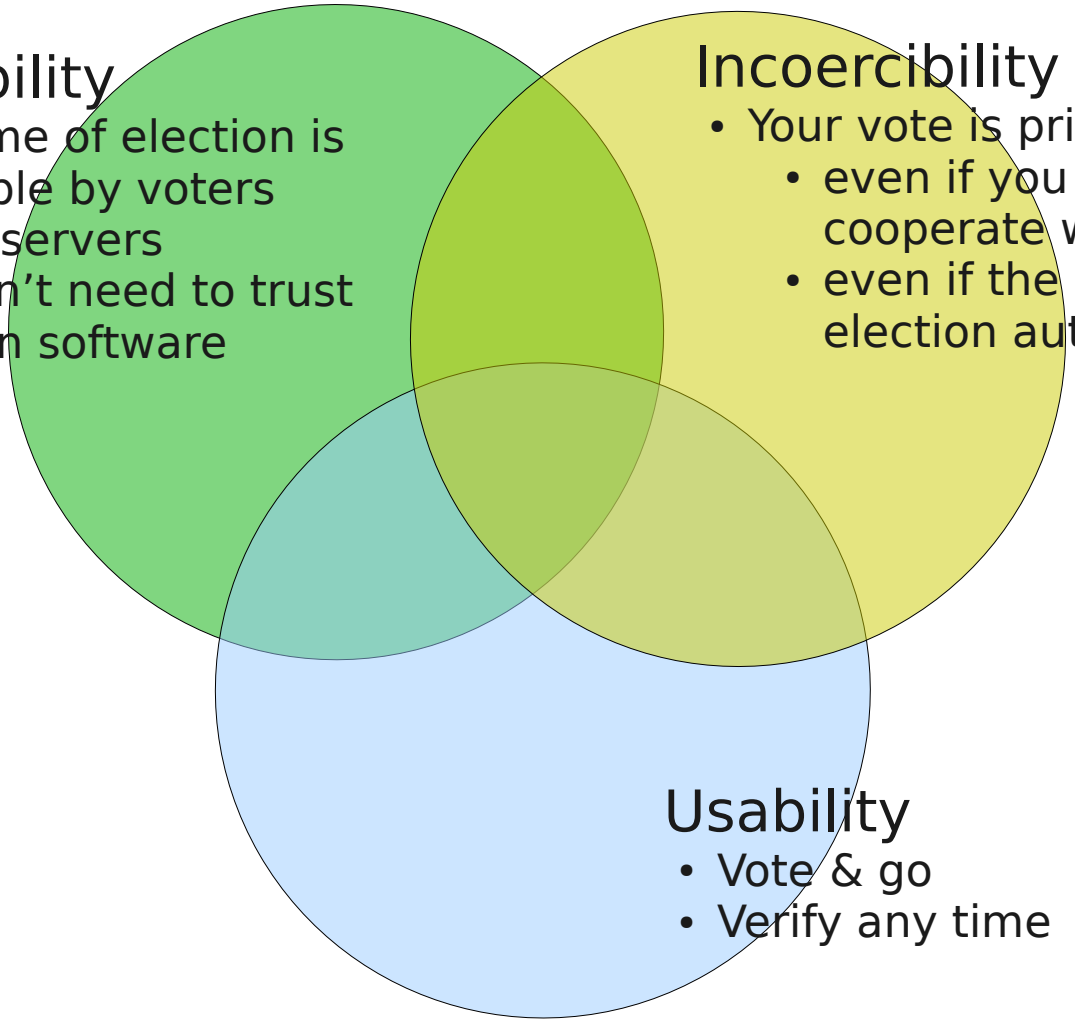
Verifiability

- Outcome of election is verifiable by voters and observers
- You don’t need to trust election software

Incoercibility

- Your vote is private
  - even if you try to cooperate with a coercer
  - even if the coercer is the election authorities

Desired properties



Verifiability

- Outcome of election is verifiable by voters and observers
- You don’t need to trust election software

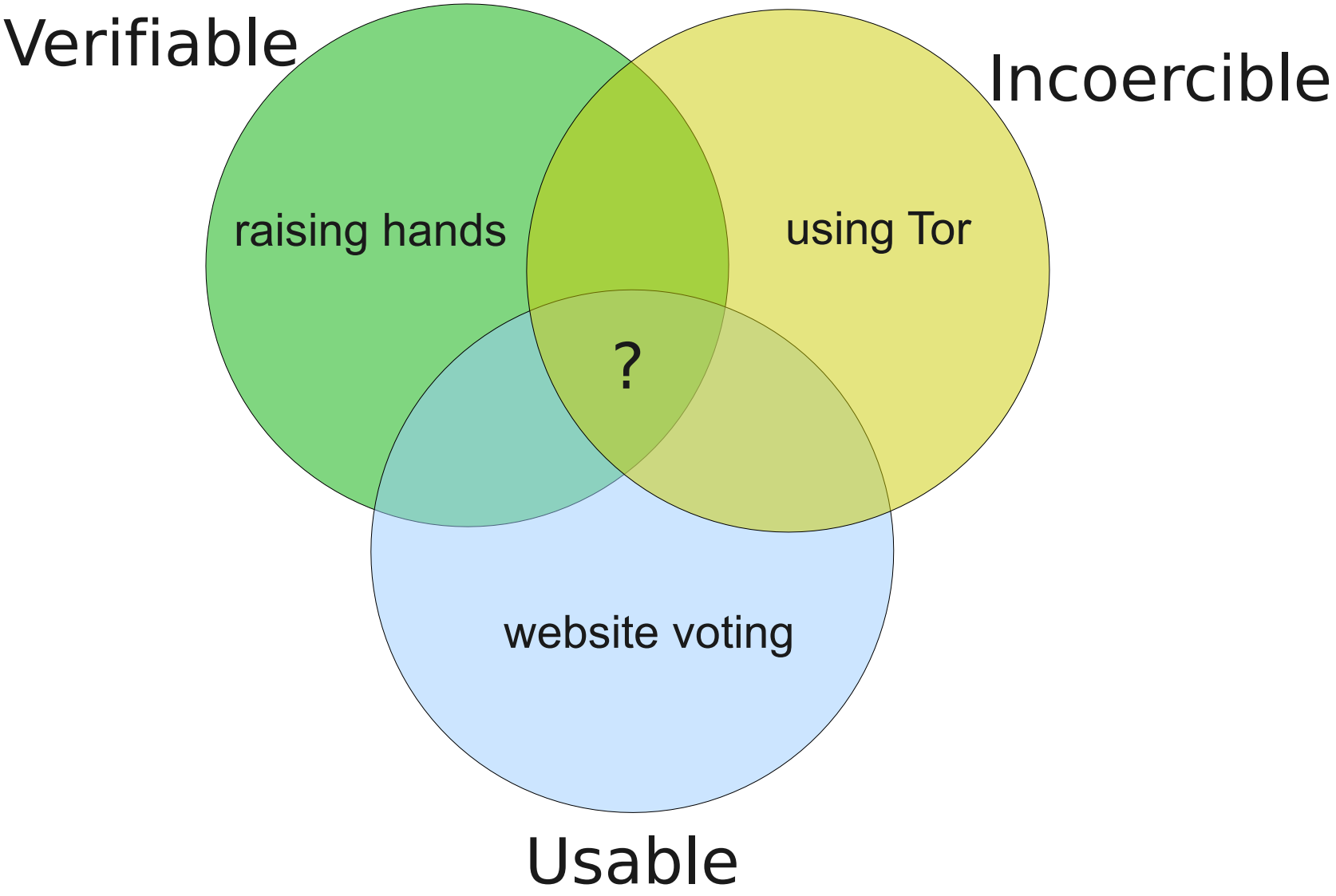
Incoercibility

- Your vote is private
  - even if you try to cooperate with a coercer
  - even if the coercer is the election authorities

Usability

- Vote & go
- Verify any time

Examples

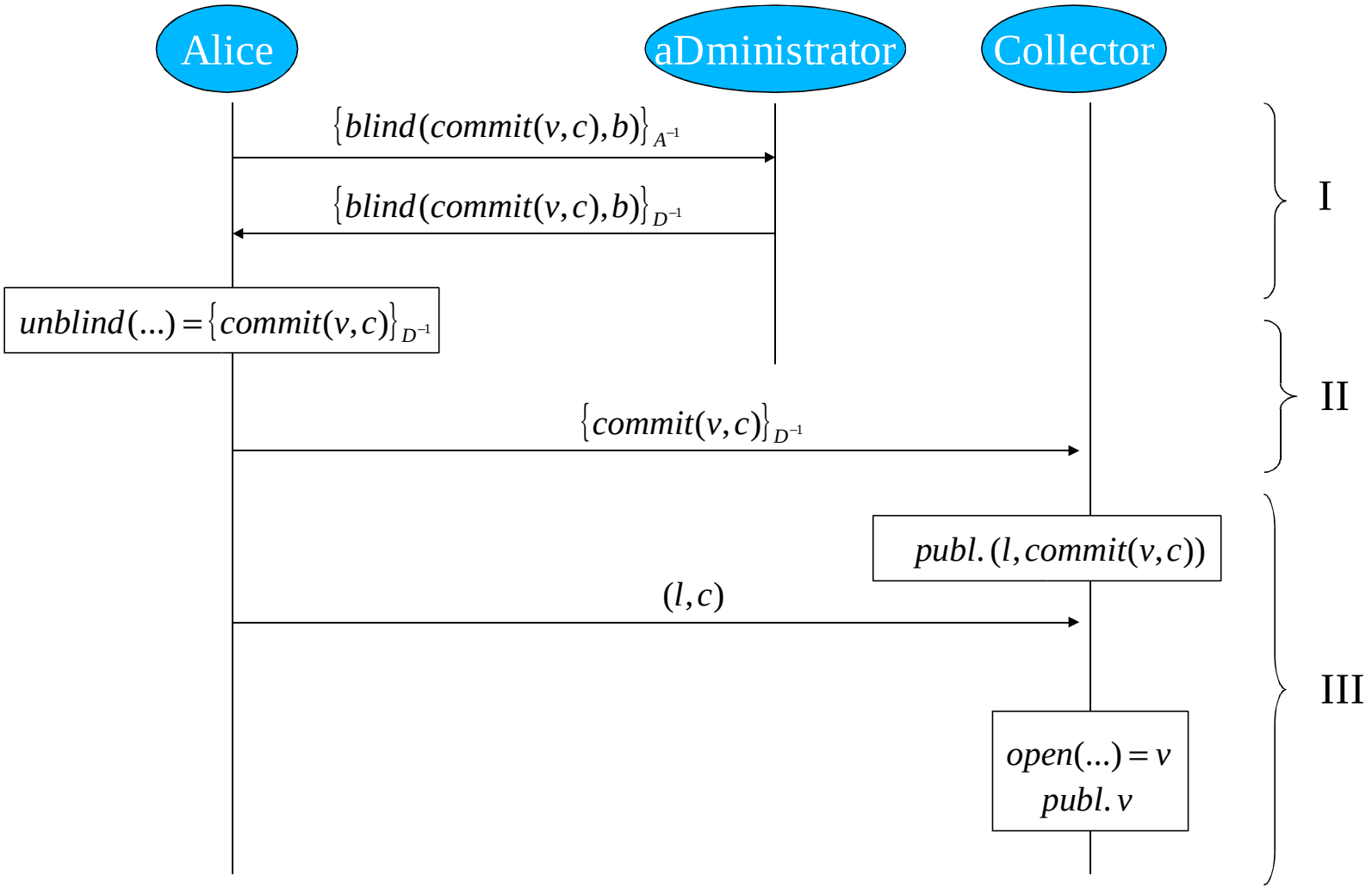

Verifiable
Incoercible
raising hands
using Tor
?
website voting
Usable

Outline

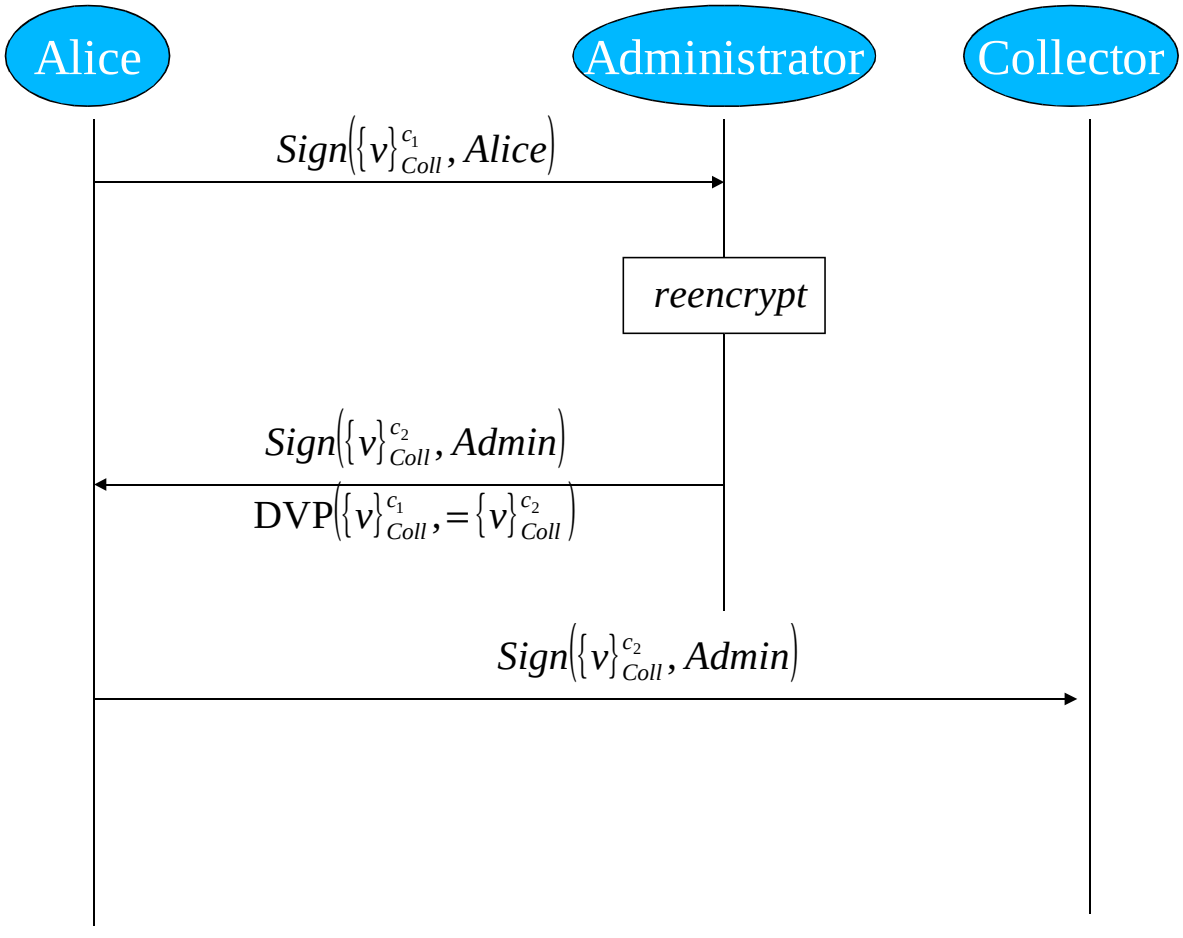
Four systems

- FOO'92
- LBDKYY'03
- Helios 2.0 2008
- JCJ/Civitas 2005–08

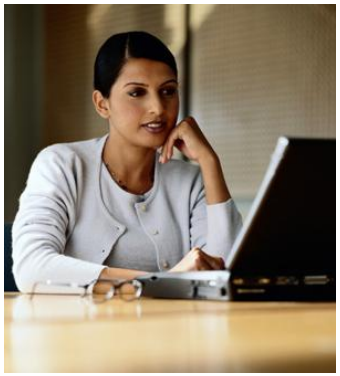
FOO 92 protocol [FujiokaOkamotoOhta92]



LBDKYY'03 protocol [LeeBoydDawsonKimYangYoo]



Helios 2.0



$\rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow$

- User prepares ballot (encrypted vote) on her computer, together with ZKPs.
- Cut-and-choose auditability provides assurance of correctness
- Not much guarantee of incoercibility on client side.

- Ballots are checked & homomorphically combined into a single encrypted outcome.
- Outcome is decrypted by threshold of talliers.
- Proof of correct decryption.

JCJ-Civitas

Credential construction

- Voters construct a secret credential through interaction with multiple registrars.
- A public part of the credential is published on the electoral register, for public scrutiny.
- Voters submit their vote with a differently randomised public part of the credential.
- Enables incoercibility and eligibility verifiability.

- Voter prepares encrypted ballot on a trusted computer, and submits it.
  - Similar to Helios 2.0, except that cut-and-choose auditability of ballot isn't necessary.
- Ballots are submitted to a re-encryption mixnet that is able to prove it correctly mixed the ballots.
- Ballots are threshold decrypted with proof of correct decryption, and counted.

Outline

Needed primitives

- encryption/decryption
  - re-encryption (re-randomisation)
  - homomorphic encryption
  - with threshold decryption
- signatures
  - blind signatures
- zero-knowledge proofs
  - proofs of encryption
  - proofs of decryption
  - designated verifier proofs

Encryption/decryption

- **decryption**

$$dec(x_{sk}, penc(pk(x_{sk}), x_{rand}, x_{text})) = x_{text}$$

- **homomorphic encryption**

$$penc(x_{pk}, y_{rand}, y_{text}) * penc(x_{pk}, z_{rand}, z_{text}) = penc(x_{pk}, y_{rand} \circ z_{rand}, y_{text} + z_{text})$$

- **re-encryption (re-randomisation)**

$$renc(x_{rand}, penc(y_{pk}, y_{rand}, y_{text})) = penc(y_{pk}, x_{rand} \circ y_{rand}, y_{text})$$

Blind signatures

$$checksign(pk(x), sign(x, y)) = ok$$
$$getmess(sign(x, y)) = y$$

$$unblind(x, sign(y, blind(x, z))) = sign(y, z)$$

Zero-knowledge proofs

A participant in a protocol can provide evidence that she performed a computation, without revealing secret data involved in the computation.
Examples:

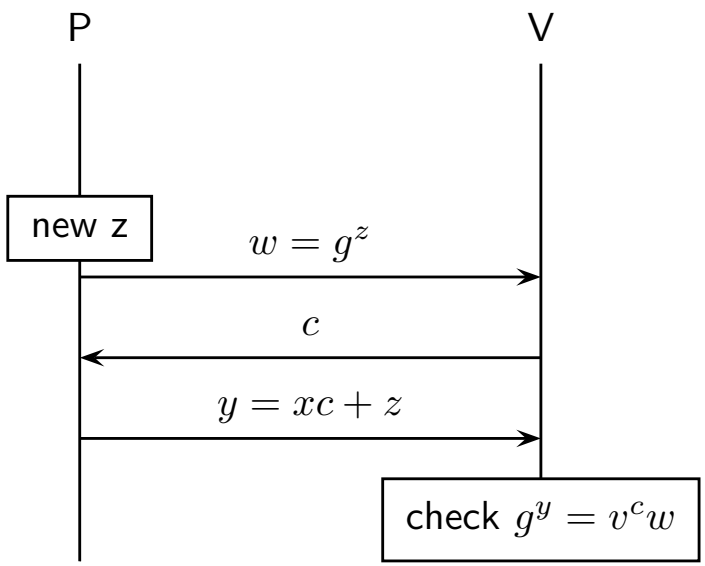
- Alice has $pk(k)$. She can make the term $enc(pk(k), r, x)$ and can prove that it has that form, without revealing r, x. She can also prove conditions on x, such as $x \in \{0, 1\}$.
- Bob has k and some encryption $enc(pk(k), r, x)$. He can decrypt to obtain x, and prove without revealing k that the decryption was performed correctly.
- Charlie has $enc(pk(k), r, x)$. He can produce $renc(r', enc(pk(k), r, x)) = enc(pk(k), r' \circ r, x)$ and prove in a way that will be convincing only to Dave that the two terms are encryptions of the same plaintext. ("Designated verifier proof of re-encryption.")

Zero-knowledge proofs

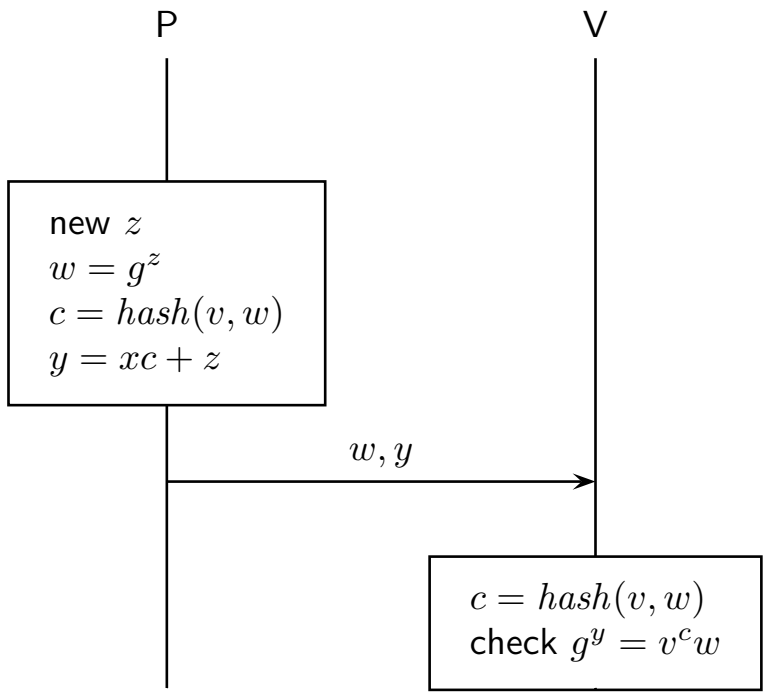
Assume the ElGamal parameters (p, q, g) and arithmetic is done mod q (exponents) and mod p (rest).

Prover P has secret x, computes $v = g^x$, sends v to verifier V. P wants to prove that v does have the form g^x for some x.

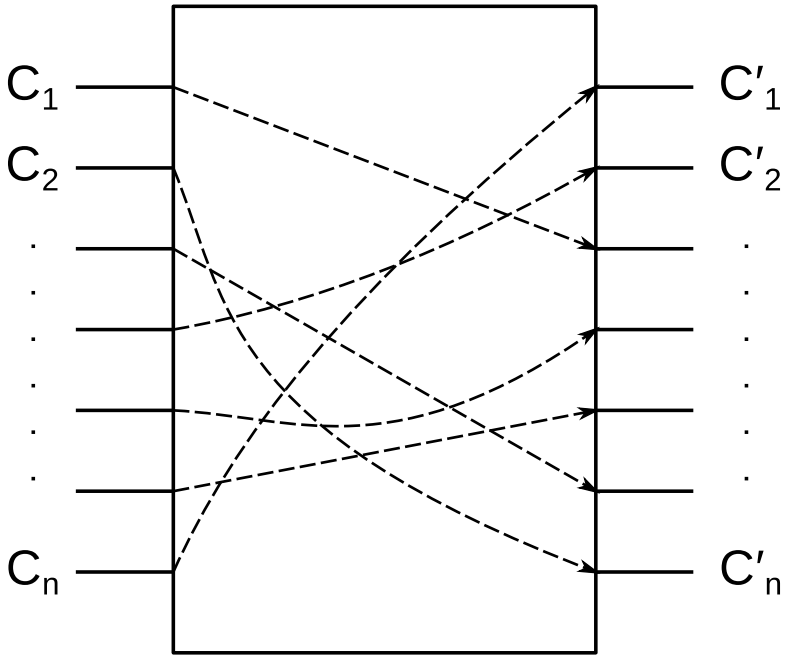
Interactive

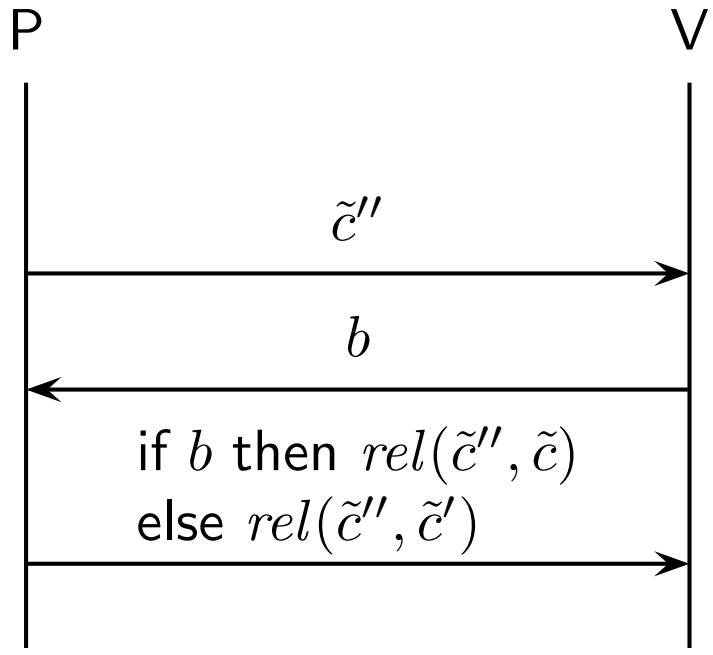


Non-interactive



Proof for re-encryption mixer





Proofs about encryption/decryption

- **Decryption**

  $$dec(x_{sk}, penc(pk(x_{sk}), x_{rand}, x_{text})) = x_{text}$$

- **Proof of encryption**

  $$checkEncPf(x_{pk}, enc, encPf(x_{pk}, x_{rand}, s, enc)) = true$$

  where $enc = penc(x_{pk}, x_{rand}, s)$

- **Proof of decryption**

  $$dec(decKey(x_{sk}, ciph), ciph) = x_{plain}$$

  where $ciph = penc(pk(x_{sk}), x_{rand}, x_{plain})$

  $$checkDecKeyPf(pk(x_{sk}), ciph, dk, decKeyPf(x_{sk}, ciph, dk)) = true$$

  where $ciph = penc(pk(x_{sk}), x_{rand}, x_{plain})$

  and $dk = decKey(x_{sk}, ciph)$

Designated verifier proof of re-encryption

Idea

Given $ciph = penc(pk(k), r, m)$, one can compute $ciph' = renc(r', ciph)$ and a proof that $ciph, ciph'$ have the same plaintext. The proof is computed involving $pk(sk_{Bob})$, in such a way that only the holder of sk_{Bob} is convinced by it.

Legitimate proof

$checkdvp(pf, x_{ciph}, renc(x_{rand}, x_{ciph}), x_{pkv}) = ok$

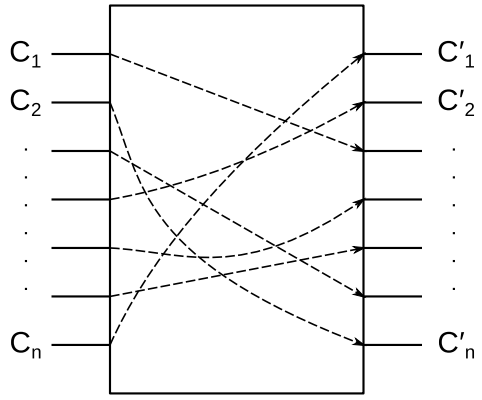
where $pf = dvp(x_{ciph}, renc(x_{rand}, x_{ciph}), x_{rand}, x_{pkv})$

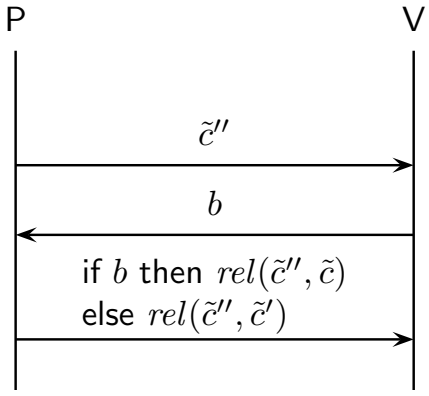
Fake proof

$checkdvp(pf, x_{rand}, y_{rand}, pk(x_{skv})) = ok$

where $pf = dvp(x_{ciph}, y_{ciph}, x_{rand}, x_{skv})$

Proof for re-encryption mixer





For each permutation π on $\{1, \ldots, n\}$:

$$checkMix(mixPf(x_1, \ldots, x_n, ciph_1, \ldots, ciph_n, z_1, \ldots, z_n), x_1, \ldots, x_n, ciph_1, \ldots, ciph_n) = ok$$

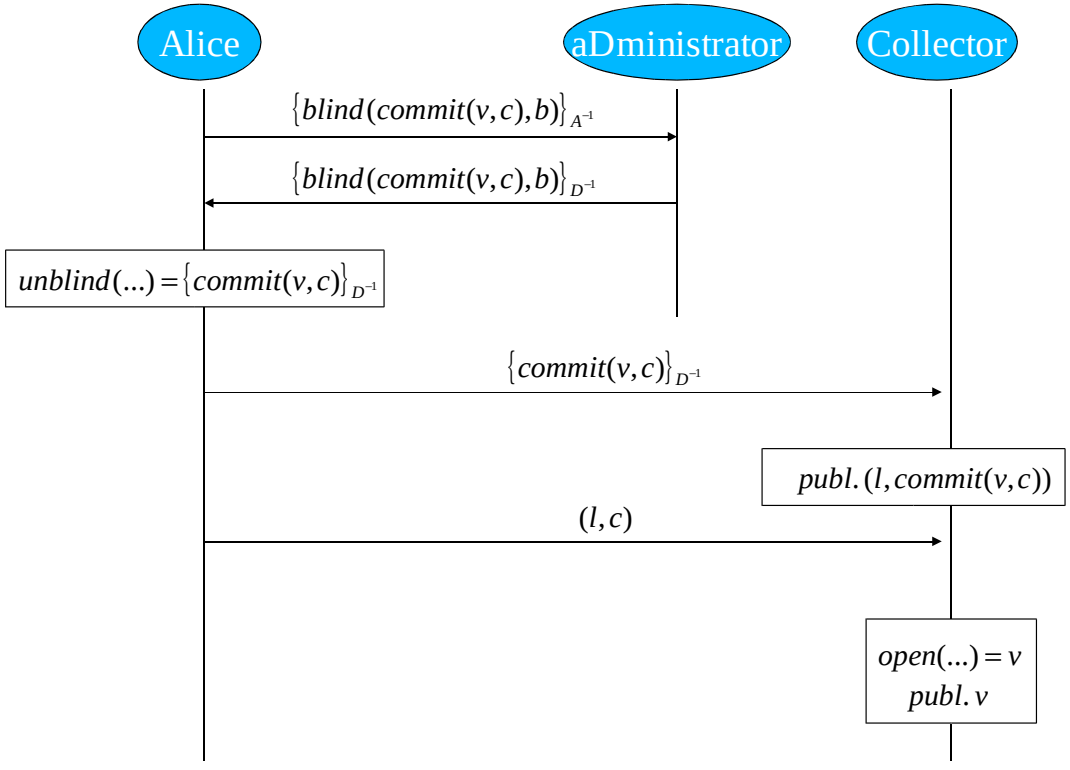
where $ciph_i = renc(z_i, x_{\pi(i)})$

Outline

Coding protocols as processes

Example ([FOO'92]):

```
processV =
  new b; new c;
  let bcv = blind(commit(v,c),b)
  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)).
```



Modelling Helios 2.0: processes

- *Voting process specification* $\langle V, A \rangle$
- *Voting process* $\mathrm{VP}_n(s_1, \ldots, s_n) = A[V_1 \mid \cdots \mid V_n]$ models the protocol with n voters casting votes for candidates $s_1, \ldots, s_n$.

Definition

The voting process specification $\langle V_{\mathsf{helios}}, A_{\mathsf{helios}} \rangle$ is defined where

$$
\begin{aligned}
V_{\mathsf{helios}} &\triangleq d(x_{\mathsf{pid}}).\ \overline{d}\langle v \rangle.\ d(x_{\mathsf{ballot}}).\ d(x_{\mathsf{ballotpf}}).\overline{c}\langle (\!| x_{\mathsf{pid}}, x_{\mathsf{ballot}}, x_{\mathsf{ballotpf}} |\!) \rangle \\
A_{\mathsf{helios}}[\_] &\triangleq \nu sk, d.\ \left(\overline{c}\langle \mathsf{pk}(sk) \rangle \mid (!\nu pid.\ \overline{d}\langle pid \rangle) \mid (!B) \mid T \mid \_\ \right) \\
B &\triangleq \nu m.\ d(x_{\mathsf{vote}}).\overline{d}\langle \mathsf{penc}(\mathsf{pk}(sk), m, x_{\mathsf{vote}}) \rangle. \\
&\qquad \overline{d}\langle \mathsf{ballotPf}(\mathsf{pk}(sk), m, x_{\mathsf{vote}}, \mathsf{penc}(\mathsf{pk}(sk), m, x_{\mathsf{vote}})) \rangle \\
T &\triangleq c(x_{\mathsf{tally}}). \\
&\qquad \overline{c}\langle (\!| \mathsf{decKey}(sk, x_{\mathsf{tally}}), \mathsf{decKeyPf}(sk, x_{\mathsf{tally}}, \mathsf{decKey}(sk, x_{\mathsf{tally}})) |\!) \rangle
\end{aligned}
$$

Modelling trusted & untrusted components

Depending on the property to be analysed, some components are required to be trusted & some not.

If it is	Then it is
Required to be trusted	Coded as a process
Auditable	Coded as a process
Not required to be trusted	Not coded as a process. The attacker can do what it wants.

Outline

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

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

Proving equivalences with ProVerif

- ProVerif can prove that a biprocess is "uniform under reductions", meaning that, in every output $out(c, choice[M, N])$ on a public channel c, the terms M, N are indistinguishable.
- Uniformity under reductions is stronger than observational equivalence. For example,

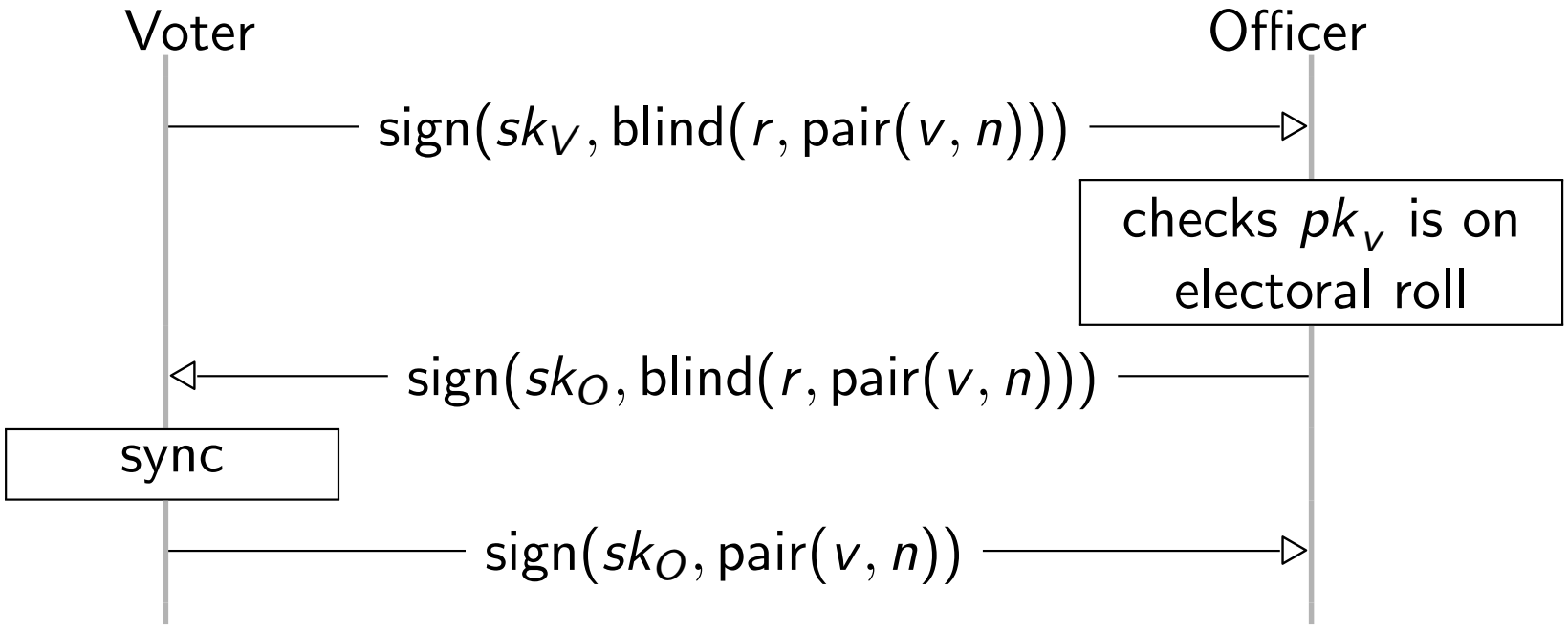
$$out(c, choice[a, b]) \mid out(c, choice[b, a])$$

  satisfies observational equivalence but is not uniform under reductions.
- This is a problem for verifying privacy-type properties of voting systems.

Ballot secrecy in electronic voting

Consider a simple voting protocol that relies on *blind signatures*. Such signatures are modelled by the equation

$$\mathsf{unblind}(x, \mathsf{sign}(y, \mathsf{blind}(x, z))) = \mathsf{sign}(y, z)$$



$$
\begin{aligned}
P(sk, v) = \; & \nu n.\nu r.let\ bvn = blind(r, pair(v, n))\ in \\
& \overline{c}\langle pair(pk(sk), sign(sk, bvn))\rangle. \\
& c(x).if\ checksign(pk_O, x) = true\ then \\
& if\ getmsg(x) = bvn\ then \\
& sync. \\
& \overline{c}\langle unblind(r, x)\rangle
\end{aligned}
$$

$$P_{sync}(sk_A, ch[a,b]) \mid P_{sync}(sk_B, ch[b,a]) \qquad \text{sat o.e.}$$

$$(P_1(sk_A, ch[a,b]) \mid P_1(sk_B, ch[b,a])) \,;\; (P_2(sk_A, ch[a,b]) \mid P_2(sk_B, ch[b,a])) \qquad \text{sat o.e.}$$

$$(P_1(sk_A, ch[a,b]) \mid P_1(sk_B, ch[b,a])) \,;\; (P_2(ch[sk_A, sk_B], a) \mid P_2(ch[sk_B, sk_A], b)) \qquad \text{sat o.e.}$$

$$P_1(sk_A, ch[a,b]); P_1(sk_B, ch[a,b]); P_2(ch[sk_A, sk_B], a); P_2(ch[sk_B, sk_A], b) \qquad \text{sat o.e.}$$

provided $P_i(sk, v)$

- doesn't block
- doesn't share secrets with its counterpart

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
$P \mid P$
$\nu n.P$
$u(x).P$
$\overline{u}\langle M\rangle.P$
if $M = N$ then P else P
$!P$
$\ldots$

P^{ch} in terms of P

- $0^{ch} = 0$
- $(P \mid Q)^{ch} = P^{ch} \mid Q^{ch}$
- $(\nu n.P)^{ch} = \nu n.\overline{ch}\langle n\rangle.P^{ch}$
- $(u(x).P)^{ch} = u(x).\overline{ch}\langle x\rangle.P^{ch}$
- $(\overline{u}\langle M\rangle.P)^{ch} = \overline{u}\langle M\rangle.P^{ch}$
- $\ldots$

We denote by $P^{\backslash \overline{chc}\langle\cdot\rangle}$ the process $\nu chc.(P \mid !chc(x))$.

Lemma: $(P^{ch})^{\backslash \overline{chc}\langle\cdot\rangle} \approx_\ell P$

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

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

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

- exhibiting V'
- showing that $V'\backslash\overline{chc}\langle\cdot\rangle \approx_\ell V_A\{^a/_v\}$
- 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

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

- $C[V']^{\backslash \overline{chc}\langle\cdot\rangle} \approx_\ell V_A\{a/v\}$,
- $S[C[V_A\{?/v\}^{c_1,c_2}] \mid V_B\{a/v\}] \approx_\ell S[C[V'] \mid V_B\{c/v\}]$.

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 is coercion-resistant

$\Downarrow$

VP is receipt-free

$\Downarrow$

VP respects privacy

Outline

Election verifiability

Individual verifiability

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

Universal verifiability

Anyone can check that the declared outcome corresponds to the ballot collection.

Eligibility verifiability

Anyone can check that only eligible votes are included in the ballot collection.

Remark

- Verifiability $\neq$ correctness

Individual and universal verifiability

Individual test

We require a test

$$\Phi^{IV}(\text{my\_vote}, \text{my\_data}, \text{bb\_entry})$$

that a voter can apply after the election. The test succeeds iff the bulletin board entry corresponds to the voter's vote and data.

Universal test

We require a test

$$\Phi^{UV}(\text{decl\_outcome}, \text{bb\_entries}, \text{pf})$$

that an observer can apply after the election. The test succeeds iff the declared outcome is correct w.r.t. the bb entries and the proof.

Individual and universal verifiability

$\Phi^{IV}(\text{my\_vote}, \text{my\_data}, \text{bb\_entry})$ $\quad$ $\Phi^{UV}(\text{decl\_outcome}, \text{bb\_entries}, \text{pf})$

Acceptability conditions for Φ^{IV} and Φ^{UV}

Soundness, For all BBs:

$$\forall i, j. \quad \Phi^{IV}(v_i, r_i, y) \wedge \Phi^{IV}(v_j, r_j, y) \Rightarrow i = j \tag{1}$$

$$\Phi^{UV}(\tilde{v}, \tilde{y}, p) \wedge \Phi^{UV}(\tilde{v}', \tilde{y}, p) \Rightarrow \tilde{v} \simeq \tilde{v}' \tag{2}$$

$$\bigwedge_{1 \leq i \leq n} \Phi^{IV}(v_i, r_i, y_i) \wedge \Phi^{UV}(\tilde{v}', \tilde{y}, p) \Rightarrow \tilde{v} \simeq \tilde{v}' \tag{3}$$

Effectiveness There exists a BB s.t.

$$\bigwedge_{1 \leq i \leq n} \Phi^{IV}_i(v_i, r_i, y_i) \wedge \Phi^{UV}(\tilde{v}, \tilde{y}, p) \tag{4}$$

Helios 2.0: Verifiability

The untrusted server is assumed to publish the election data. When the protocol is executed as expected the resulting frame should have substitution σ such that

$$
\begin{aligned}
x_{pk}\sigma &= pk(sk) \\
y_i\sigma &= (\!|\, pid_i, penc(pk(sk), m_i, v_i), \\
&\qquad ballotPf(pk(sk), m_i, v_i, penc(pk(sk), m_i, v_i)) \,|\!) \\
z_{tally}\sigma &= \pi_2(y_1) * \cdots * \pi_2(y_n)\sigma \\
z_{decKey}\sigma &= decKey(sk, z_{tally})\sigma \\
z_{decKeyPf}\sigma &= decKeyPf(sk, z_{tally}, z_{decKey})\sigma
\end{aligned}
$$

$$
\begin{aligned}
\Phi^{IV} &\widehat{=} y =_E (\!|\, r_{pid}, r_{ballot}, r_{ballotpf} \,|\!) \\
\Phi^{UV} &\widehat{=} z_{tally} =_E \pi_2(y_1) * \cdots * \pi_2(y_n) \\
&\quad \wedge \textstyle\bigwedge_{i=1}^{n} (checkBallotPf(x_{pk}, \pi_2(y_i), \pi_3(y_i)) =_E true) \\
&\quad \wedge checkDecKeyPf(x_{pk}, z_{tally}, z_{decKey}, z_{decKeyPf}) =_E true \\
&\quad \wedge v_1 + \cdots + v_n =_E dec(z_{decKey}, z_{tally})
\end{aligned}
$$