A (somewhat) gentle introduction to machine-checked cryptographic proofs
Privacy and Verifiability for online voting

François Dupressoir
joint work with V Cortier, CC Drăgan, B Warinschi, P-Y Strub, B Schmidt, B Grégoire, and others
Online Voting: Some Context

» Electronic Voting is already widely used
  • Non-governmental elections and votes
  • Legally-binding decision-making (Switzerland, Estonia, …)

» Problem: attacks to privacy and integrity scale up
  • Traditional approaches (observe and audit) are insufficient

» Solution: Throw crypto in and mix
  • Prove strong privacy and integrity guarantees, under…
  • … standard cryptographic assumptions, and …
  • … simple trust assumptions.
What is Online Voting? (Syntax)

» The *election authority* sets up the election, generates the voter roll, checks eligibility, ...
  • Modelled as a **Setup** algorithm

» The *voters* cast their votes, and later may want to check them
  • Modelled as a **Vote** algorithm

» The *ballot box* receives ballots
  • Modelled as a **Valid** algorithm (+ a chunk of state)

» The *bulletin board* holds a public view of the ballots received and other verifiability evidence
  • Modelled as a **Publish** algorithm (+ a chunk of state)

» The *trustees* compute the tally from ballots held within the ballot box
  • Modelled as a **Tally** algorithm

» The *general public* may want to check the good conduct of the election
  • Modelled as a **Verify** algorithm
A Typical (secure) Online Voting System

- **Setup** generates a keypair for the election
  - Usually shared between trustees so that a threshold of them need to collaborate to decrypt

- **Vote** encrypts voter choice under the election public key, may protect the integrity of the ballot

- **Valid** typically prevents direct replay of encrypted ballots, rejects ill-formed ballots
  - May prevent revotes, …

- **Publish** typically selects a subset of information to publish
  - May publish nothing at all (no verifiability)

- **Verify** checks that tallying was performed correctly
Two Ways of Tallying

» Homomorphically
  • Vote uses (partially) homomorphic encryption
  • Tally computes homomorphically over ciphertext to get encrypted result
  • A threshold of trustees decrypt the result once they agree tallying is finished, and produce a NIZK proof of correct decryption

» Using mix-nets
  • Vote uses re-randomizable encryption
  • A network of mix-servers sequentially re-randomize the ciphertexts after shuffling them, producing NIZK proofs of correct shuffling
  • A threshold of trustees decrypt individual ballots once they agree shuffling is finished
  • Tally can then be performed publicly

» We want tally-agnostic definitions for privacy and verifiability
Defining and Proving Privacy for Online Voting
Defining Privacy I

» Long history of bad game-based definitions

» Ideally, want guarantees as strong as those given by a true Trusted Third Party (Simulation-Based Security)
   • There exists a simulator such that no adversary can distinguish the scheme from the simulator

» Simulations are really hard to deal with
Defining Privacy II

» Bernhard, Cortier, Galindo, Pereira and Warinschi (S&P 15) define BPRIV
  • Prove that, with two simpler conditions, it implies simulation-based privacy

» BPRIV is “mostly” game-based
  • Easy to manipulate and instantiate

» BPRIV is a “Left or Right” game
Defining Privacy III (BPRIV)

» **Vote** oracle is the only one that is made Left or Right
  • Voter choice is the only thing whose privacy we care to protect

» The adversary is additionally given the ability to form and cast ballots without using **Vote**
  • Models voters who may be under adversary control

» In the Left game, everything works as expected:
  • Simply run the scheme using the left input to **Vote** oracle queries

» In the Right game, things get complicated fun:
  • Maintain Ballot Boxes corresponding to both sets of inputs
  • **Publish** gives the public bulletin board produced from the Right box
  • **Tally** computes the result using the Left box,
    and *simulates* a proof that the result was computed correctly from the Right box
### Defining Privacy IV (BPRIV, a formal view)

<table>
<thead>
<tr>
<th>Oracle $\text{Ocast}(b)$</th>
<th>Oracle $\text{Otally}(\cdot)$ for $\beta = 0$</th>
<th>Oracle $\text{Otally}(\cdot)$ for $\beta = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: if $(\text{Valid}(\text{BB}_\beta, uL, b, pk))$ then</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2: $\text{BB}_0 \leftarrow \text{BB}_0 + [b]$; $\text{BB}_1 \leftarrow \text{BB}_1 + [b]$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1: $(r, II) \leftarrow \text{Tally}(\text{BB}_0, sk)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3: return $(r, II)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1: $(r, II) \leftarrow \text{Tally}(\text{BB}_0, sk)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2: $II' \leftarrow \text{Sim}(pk, \text{Publish}(\text{BB}_1, r)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3: return $(r, II')$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Oracle $\text{Oboard}(\cdot)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: return $\text{Publish}(\text{BB}_\beta)$</td>
</tr>
</tbody>
</table>

Oracle $\text{Otally}(\cdot)$ for $\beta = 0$ and $\beta = 1$.
Proving Privacy?

- 2 published attempts at proving BPRIV for Helios-like protocols
  - Both had minor issues and a significant gap

- Zero-Knowledge proofs evidence a mathematical relation between a (secret) witness and a (public) statement

- The language of valid statements should be in NP

- In our case, the statement talks about the random oracle
  - If we make it stateless, not in NP
  - If we make it stateful, need new theory

- This was never highlighted as an issue…
A New Problem

- Nobody understands cryptographic proofs
  - Hard to write, but *even harder to read*

- Formalize the proof in EasyCrypt
  - Introduces an asymmetry between proof writer and proof reader
  - Removes focus from the proof itself, and
  - Allows evaluator to focus on definitions and claims

- Key insight:
  - All crypto security notions (even simulation-based) are post-conditional equivalences between open probabilistic programs
  - Relational reasoning is well suited

- EasyCrypt allows us to dive below program logics and into semantics
We formalize BPRIV and its associated properties for Labelled MiniVoting (Bernhard et al.)

<table>
<thead>
<tr>
<th>Setup($1^λ, m$)</th>
<th>Vote($id, ℓ, v, pk$)</th>
<th>Valid($BB, uL, b, pk$)</th>
<th>Tally($BB, sk$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: $(pk, sk) ← KGen(1^λ)$</td>
<td>1: $c ← Enc(pk, ℓ, v)$</td>
<td>1: $(id, ℓ, c) ← b$</td>
<td>1: $dbb = []$</td>
</tr>
<tr>
<td>2: for $i$ in 1..$m$ do</td>
<td>2: return $(id, ℓ, c)$</td>
<td>2: $e_1 ← \forall id'. (id', ℓ, c) \notin BB$</td>
<td>2: for $i$ in 1..</td>
</tr>
<tr>
<td>3: $id ← ID$</td>
<td>3: $e_2 ← (ℓ = uL[id])$</td>
<td>3: $(id, ℓ, c) = BB[i]$</td>
<td>3: $r ← \rho(dbb)$</td>
</tr>
<tr>
<td>4: $uL[id] ← Flabel(id)$</td>
<td>4: $e_3 ← \text{ValidInd}(b, pk)$</td>
<td>4: $dbb[i] ← (id, Dec(sk, ℓ, c))$</td>
<td>4: $pbb ← \text{Publish}(BB)$</td>
</tr>
<tr>
<td>5: return $(pk, sk, uL)$</td>
<td>5: return $(e_1 \land e_2 \land e_3)$</td>
<td>5: $r ← \rho(dbb)$</td>
<td>6: $\Pi ← P((pk, pbb, r), (sk, BB))$</td>
</tr>
</tbody>
</table>

» **Tally** is trusted, which we really do not want in practice

» We generalize over previous definitions by parameterizing the scheme

- **Flabel, ValidInd, ρ, R**

#universityofsurrey
By verifiably secure refinement, we transfer the security of “Labelled MiniVoting” to:

- Mixnet-based Helios (Helios v3-mix)
- Homomorphic Helios (Helios v3-hom, Helios v4)

By observational equivalence, we further transfer the privacy result to a previously unproved optimized version of Helios v4.

By verifiable instantiation, we obtain machine-checked privacy proofs for over 500 variants of Helios.
Verification Effort

» About 1 person-year from start of project to “final” qed
  • Includes false starts, time for RA to learn both the crypto and formal tools
  • Roughly 1,500 lines of definitions (includes named variants)
  • Roughly 14,000 lines of proof (includes named variants)
  • Unnamed variants are automatically generated – ~150LoD, ~500LoP each

» Initial proof for Labelled MiniVoting obtained about 75% of the effort in
  • But later iterations needed to extend treatment of proofs over relations that include random oracles

» Identified a missing assumption in published proofs of Labelled MiniVoting
  • Does not affect practical security, since it is discharged on concrete instantiations
Defining and Proving Verifiability for Online Voting
We want verifiability
  • If something goes wrong, anywhere, we want to know it

We want it with minimal trust assumptions
  • Force at least two parties to collude to subvert the election without detection

End-to-End verifiability relies on:
  • Individual Verifiability: individual voters should be able to verify that their vote was both cast as intended, and recorded as cast
  • Universal Verifiability: anyone should be able to verify that all votes were counted as recorded

But end-to-end verifiability does not prevent ballot stuffing
  • A malicious ballot box can just stick ballots in for voters who haven’t voted
Belenios (Cortier et al. 2014)

- To prevent ballot stuffing, voters need cryptographic credentials
- A registrar manages a mapping from eligible voters to their public keys
  - The registrar does not maintain the voter roll – that *must* be trusted
- Signed ballots are signed by voters before being cast
  - A malicious ballot box cannot stuff ballots, as it doesn’t have the voters’ signing keys
- For privacy, signatures must be stripped before tallying
- Cortier et al. provide proofs of privacy and verifiability
Machine-Checked Privacy for Belenios

Expectation:
1. Add registration and signing in Helios definitions and proofs
2. Run EasyCrypt
3. Minimally fiddle with proof
4. Profit

Reality:
1. Add registration and signing in Helios definitions and proofs
2. Run EasyCrypt
3. ??
4. Wat?
Privacy for Belenios: No Dynamic Corruption

» Dynamic corruption allows adversary to replay an honest ballot to learn the vote it contains

\[
\begin{align*}
\beta &= 0 \\
BB_0 &= [(id, b_0), (id, b_0)] \\
(v_0, \pi) &\leftarrow Tally(BB_0) \\
\text{Adversary sees board } BB_0, \\
& \text{result } v_0, \text{ and real proof } \pi
\end{align*}
\]

\[
\begin{align*}
\beta &= 1 \\
BB_1 &= [(id, b_1), (id, b_1)] \\
BB_0 &= [(id, b_0), (id, b_1)] \\
(v_1, \pi) &\leftarrow Tally(BB_0) \\
\text{Adversary sees board } BB_1, \\
& \text{result } v_1, \text{ and sim. proof } \pi'
\end{align*}
\]

» Helios accidentally avoids this issue by preventing replay of ballots
  • Which was put in place to stop an actual attack on privacy
Privacy for Belenios: Trust and the Registrar

» A dishonest registrar can give invalid credentials to all voters but one
  • Tally reveals that voter’s preferences

» Is this an attack?
  • Yes

» Should we care about it?
  • Yes

» Why? Both attacks are artificial: the adversary truly learns nothing in practice…
Privacy for Belenios – The Big Problem

» Current definitions and natural extensions are not robust
  • Consideration of elements usually left out of scope is needed

» In a follow-up, Cortier and Lallemand prove that all current definitions of privacy imply individual verifiability

» We can only get proofs in much weaker models than those we want

» There may be actual attacks we are missing because of “silly” definitional issues

» We’re not solving this here: we just accept a weaker definition and move on
Strong Verifiability for Belenios

» If the adversary:
  • Controls either the registrar or the ballot box,
  • Knows the election private key, and
  • Can corrupt individual voters statically.

» The final tally corresponds to the tally computed over:
  • The votes of all honest voters who perform individual verifiability checks,
  • A subset of the votes cast by honest voters who did not check,
  • At most as many corrupted votes as there were corrupted voters.

» In practice, this is very strong as it gets
  • Gives precise bounds on the distance between final result and actual result
  • The adversary does not know who will check; statistical arguments can give tighter bounds
Machine-Checked Verifiability for Belenios

» Really nothing to report; some extensions and clarifications to Cortier et al.’s result:
   • Give the private election key to the adversary
   • Refine what it means for a voter to have checked their ballot (in case of revotes)
   • When registrar is dishonest, even honestly generated ballots may be invalid and cannot be counted

» The proofs are “straightforward formalizations”

» Could further refine to allow checking on intermediate bulleting boards when they are published

<table>
<thead>
<tr>
<th>Belenios</th>
<th>LoC</th>
<th>Ver. Time (s)</th>
<th>Code Sim. (%)</th>
<th>Dev. Effort (PW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>General Concepts</td>
<td>5936</td>
<td>348</td>
<td>55% Helios</td>
<td>4</td>
</tr>
<tr>
<td>Privacy</td>
<td>2700</td>
<td>238</td>
<td>75% Helios</td>
<td>2</td>
</tr>
<tr>
<td>Verifiability</td>
<td>14590</td>
<td>1523</td>
<td>-</td>
<td>20</td>
</tr>
<tr>
<td>Variants</td>
<td>47030</td>
<td>3965</td>
<td>95% Belenios</td>
<td>1</td>
</tr>
</tbody>
</table>
Next Steps in Machine-Checked Cryptographic Proofs
Our proofs do not cover the primitives:
- Mix-nets are assumed to be perfect obliviously permuting decryption oracles
- Zero-Knowledge proofs are taken as assumptions
  - Including those whose statements talk about random oracles
- Encrypt+PoK is taken as non-malleable encryption

These proofs are fun
- Fun number theory
- Interesting proof techniques, where simulators can rerun adversaries with fixed randomness
- Zero-Knowledge is still not very well understood in terms of proofs, composition
- Even though these things are conceptually simple, they involve interactive systems
Interactive systems are also increasingly used by the crypto community for compositional security
  • Constructive Cryptography
  • Universal Composability

Having proof tools that support them will be crucial in scaling machine-checked crypto up to larger constructions

Ideas from distributed system verification could be looked into
» Machine-checked crypto is costly
» But worth it for select applications where trust in the system is paramount
  • Standards, voting, e-government, privacy, …

» Definitions of privacy for electronic voting are brittle and inadequate

» PL and PV can still contribute to machine-checked crypto
  • Weird rewinding semantics
  • (Relational) Semantics for interactive open probabilistic programs