NTNU

Norwegian University of
Science and Technology

Trondheim – Gjøvik – Ålesund

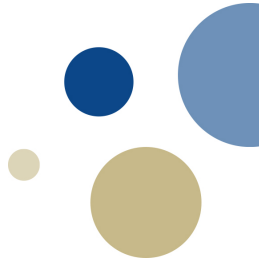
Formal Verification of Cryptographic Voting Schemes

Morten Solberg

Department of mathematical sciences, NTNU

January 30th 2020

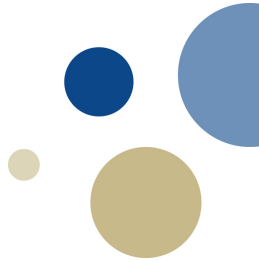
Electronic voting vs. traditional Voting



Traditional voting has some limitations:

- Relies on trust in election officials
- Counting errors
- Accessibility
- ...

Electronic voting vs. traditional Voting



Traditional voting has some limitations:

- Relies on trust in election officials
- Counting errors
- Accessibility
- ...

Electronic voting may benefit from:

- Increased trust by distributed tasks
- Computer aided counting
- Public verification
- ...

How do we know our voting scheme is secure?

We apply the standard method of provable security, with reductionist proofs.

How do we know our voting scheme is secure?

We apply the standard method of provable security, with reductionist proofs.

But what does "secure voting scheme" really mean? There are several security demands we want to meet:

- Privacy (ballot privacy, receipt-freeness, coercion resistance)
- Verifiability (cast-as-intended, counted-as-cast)
- Robustness
- Integrity
- ...

How do we know our voting scheme is secure?

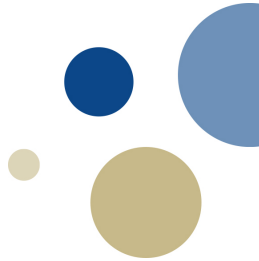
We apply the standard method of provable security, with reductionist proofs.

But what does "secure voting scheme" really mean? There are several security demands we want to meet:

- Privacy (ballot privacy, receipt-freeness, coercion resistance)
- Verifiability (cast-as-intended, counted-as-cast)
- Robustness
- Integrity
- ...

In summary: define what "secure" means and provide a security proof.

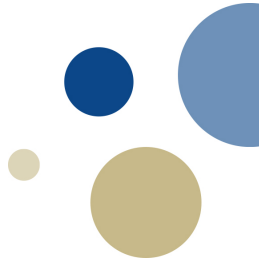
How do we know our proof is correct?



We use computer-aided proof assistants to formally verify our security proofs.

Proofs are transformed (semi-automatically) to simple logic statements, which are checked line by line by a *trusted computing base*.

How do we know our proof is correct?

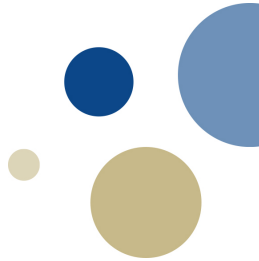


We use computer-aided proof assistants to formally verify our security proofs.

Proofs are transformed (semi-automatically) to simple logic statements, which are checked line by line by a *trusted computing base*.

However: there may be bugs (arbitrary prime $= 1$).

EasyCrypt



EasyCrypt is a proof assistant for verifying constructions in the computational model.

Security goals and hardness assumptions are modeled as probabilistic algorithms (*games*).

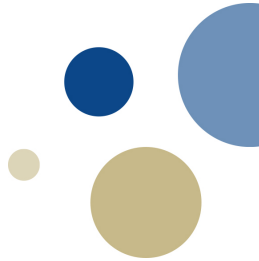
EasyCrypt has been used to verify the ballot privacy of Helios and Belenios (among others).

Goal: create a "tool-box" of formally modeled security notions and voting schemes.

A simple EasyCrypt example: Traditional voting

```
(*****************************************************************************************************)
(**** Model the voting scheme.                                                                  ****)
(*****************************************************************************************************)
module type VS = {
  proc setup()          : unit
  proc valid(id:V)      : bool
  proc cast(id:V, b:B)  : receipt
  proc tally(bb:B list) : B list
}.

module BBvoting : VS = {
  proc setup() = {}

  proc valid(id:V) : bool = {
    var ok;
    if (id \in GV.ids) {
      ok     <- false;
    } else {
      ok     <- true;
      GV.ids <- GV.ids ++ [id];
    }
    return ok;
  }

  proc cast(id:V, b:B) : receipt = {
    var ok;
    ok <@ valid(id);
    if (ok) { GV.bb <- GV.bb ++ [b]; }
    return ok;
  }

  proc tally(bb:B list) : B list = {
    var bb_n;
    bb_n <$ duniform (allperms bb);
    return bb_n;
  }
}.
```

Next steps



- Model Selene in EasyCrypt
- Define a ballot privacy model which accommodates Selene
- Prove the ballot privacy property of Selene
- Verify the proof in EasyCrypt

Thank you for your attention! Questions?