

E-Voting Privacy - Decentralized voting

Panagiotis Grontas

04/05/2023

NTUA -Advanced topics in cryptography (2022-2023)

Privacy

Intuition

Nobody can learn the vote cast by a particular voter.

- Secrecy in voting differs from secrecy in messaging
- Ballot privacy is not absolute
- The result leaks information
 - In a unanimous vote, everyone knows how everyone voted
 - In an all-but-one vote, the one that differs knows how everyone else voted
 - The result also yields a probability of a particular vote
 - Important for small voting populations

Game-based definitions for ballot secrecy

Threat model: \mathcal{A} corrupts voters. TA is honest!

Intuition: Indistinguishability games for cryptographic secrecy

Instead of distinguishing between message encryptions, \mathcal{A} tries to distinguish between different voting scenarios.

Workflow:

- \mathcal{C} setups the protocol and begins with an empty BB
- \mathcal{A} corrupts voters and casts ballots on their behalf
- For an honest voter, \mathcal{A} selects 2 choices and hands them to \mathcal{C}
- \mathcal{C} flips a coin and selects one of the choices
- \mathcal{C} creates and casts the respective ballot
- \mathcal{A} casts more corrupted ballots
- The \mathcal{A} must guess the coin

This must be combined with verifiability and apply to all possible voting rules (i.e. result functions) - voting rules.

Impact of voting rules - A toy example [CS13]

Assume the majority result function and candidates $\{a, b\}$

The candidate with majority is declared the winner

In case of a tie: a is the winner.

There are two honest voters and one corrupt voter

In scenario 1: $\{(i_0, a)(i_1, a)\}$ winner: a

In scenario 2: $\{(i_0, a)(i_1, b)\}$ winner: a

\mathcal{A} can distinguish the scenario by casting a ballot for b

Ballot independence

A voter **cannot repost** an exact replica of another voter's ballot in the BB

Assume an election with 3 voters, where V_1, V_2 are honest and V_3 is corrupt.

$$\mathbf{b}_1 = (1, (R_1, S_1), \pi_{\text{Vote},1})$$

$$\mathbf{b}_2 = (2, (R_2, S_2), \pi_{\text{Vote},2})$$

The voter replays an exact replica of \mathbf{b}_1

$$\mathbf{b}_3 = (3, (R_1, S_1), \pi_{\text{Vote},1})$$

The candidate that receives more than 2 votes is the one preferred by V_1

Alternative implementation: Use malleability and construct a different ballot for the same vote

Countermeasures:

- Ballot weeding
- Strong Fiat-Shamir Heuristic to counter encryption malleability
Recall: Enc + PoK provides non malleability
- Add voter id to the hash function

Despite being naive such attacks provide a sanity check for voting systems

Goal

A private protocol does not leak any more information than what is leaked from the tally.

Main idea

\mathcal{A} : Tries to distinguish between two worlds by having access to:

- The 'real' BB that contains honest and adversarial ballots. The adversary sees the real result.
- A 'fake' BB. The adversary sees the real results. Proofs of correctness are simulated.

Overview of BPRIV

\mathcal{A} has access to the following oracles:

- **Board**(b): Retrieve the contents of the BB_b (**Publish**)
- **Vote**(v_0, v_1): Select two votes and post the corresponding ballots to BB_0, BB_1 respectively - ballots are honestly created
- **Cast**(b): Cast a ballot to both BB - ballot is adversarially created
- **Tally**(b): Obtain the result of BB_0 . Yield real or simulated proofs

\mathcal{A} can call the oracles **Board**, **Vote**, **Cast** at will

\mathcal{A} can call **Tally** only once

\mathcal{A} must guess which BB is presented

The BPRIV definition

Algorithm 1: $\text{BPRIV}_{\mathcal{A}, \text{VS}}^b$

Input : security parameter λ

Output: $\{0, 1\}$

$(pk_{\text{TA}}, sk_{\text{TA}}) \leftarrow \text{VS.Setup}(1^\lambda)$

$\text{CS} \leftarrow \mathcal{A}()$

$b' \leftarrow \mathcal{A}^{\text{Vote, Cast, Tally, Board}}(pk_{\text{TA}})$

return $b = b'$

BPRIV

A voting system VS supports ballot privacy if there exists a simulator for \mathcal{S} such that $\forall \text{PPT } \mathcal{A}$:

$$|\Pr[\text{BPRIV}_{\mathcal{A}, \text{VS}}^0(\lambda) = 1] - \Pr[\text{BPRIV}_{\mathcal{A}, \text{VS}}^1(\lambda) = 1]| = \text{negl}(\lambda)$$

Algorithm 2: Oracles for BPRIV definition

```
Oracle Vote( $i, vt_0, vt_1$ )
┌
   $b_0 := \text{Vote}(i, vt_0)$ 
   $b_1 := \text{Vote}(i, vt_1)$ 
  if Valid( $b_0, BB_0$ ) AND Valid( $b_1, BB_1$ ) then
    ┌
       $BB_0 \leftarrow b_0$ 
       $BB_1 \leftarrow b_1$ 
    └
  else
    └ return  $\perp$ 
```

```
Oracle Cast( $i, b, b$ )
┌
  if Valid( $b, BB_b$ ) then
    ┌
       $BB_0 \leftarrow b$ 
       $BB_1 \leftarrow b$ 
    └
  else
    └ return  $\perp$ 
```

```
Oracle Tally( $b$ )
┌
  if  $b = 0$  then
    ┌
       $(T, \pi_T) := \text{Tally}(sk_{TA}, CS, BB_0)$ 
      return  $(T, \pi_T)$ 
    └
  else
    ┌
       $(T, \pi_T) := \text{Tally}(sk_{TA}, CS, BB_0)$ 
       $\pi'_T := \mathcal{S}(sk_{TA}, CS, BB_0, BB_1, T)$ 
      return  $(T, \pi'_T)$ 
    └
```

```
Oracle Board( $b$ )
┌
  return Publish( $BB_b$ )
```

Observation

The visible BB consists of tuples (i, \mathbf{b}, \cdot) posted using **Cast**, **Vote**

If the tuple originates from **Vote**, the challenger can internally map it to a tuple $(i, \mathbf{b}_0, \mathbf{vt}_0, \mathbf{b}_1, \mathbf{vt}_1)$

If the tuple originates from **Cast**, then it can be mapped to $(i, \mathbf{b}, \perp, \mathbf{b}, \perp)$

Proof overview

A sequence of games beginning from BPRIV^0 and ending to BPRIV^1 with indistinguishable differences for \mathcal{A}

Game₀: BPRIV⁰ - \mathcal{A} sees BB_0 and the real result of the tally

Game₁: The tally proof in BPRIV⁰ is simulated as

$\pi'_T = \text{Simulate}(g, \text{pk}, R, Sg^{-t}, c)$ where $c \leftarrow \mathbb{Z}_q$.

Based on the ZK property of π_T , \mathcal{A} has a negligible advantage in distinguishing between **Game₀**, **Game₁**.

$\{\text{Game}_{2,i}\}_{i=1}^{n_{\text{Hon}}}$

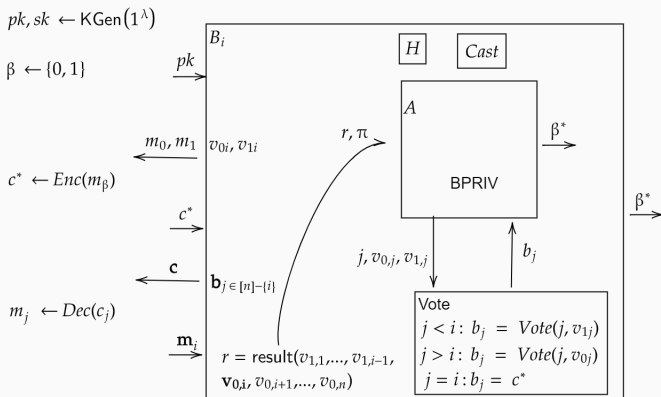
For all ballots cast using **Vote** (challenger entries $(i, \mathbf{b}_0, \mathbf{vt}_0, \mathbf{b}_1, \mathbf{vt}_1)$) replace \mathbf{b}_{0i} in BB_0 with \mathbf{b}_{1i} from BB_1 .

\mathcal{A} notices this with negligible probability due to the **NM – CPA** property of $\text{Enc} + \text{PoK}$.

Game_{2, n_{Hon}} is BPRIV¹

NM – CPA \Rightarrow $\text{Game}_{2,i} \equiv \text{Game}_{2,i-1}$

Definition of NM – CPA from [BS]



Everlasting Privacy

Motivation

- Ballot secrecy is provided through encryption schemes
- Protection relies on computational hardness assumptions
- What if these assumptions are broken?

- Vote contents might be useful to a future oppressive government
- The need for verifiability makes election data publicly available
- But such a regime might also use insider information
- This threat might constitute an indirect coercion attempt

Everlasting Privacy [MN06]

Perfectly hiding commitment schemes

- Ballot: Perfectly hiding commitment of vote
- For counting: openings are required
- How do voters post the openings?
 - Encrypt the openings
 - Secret sharing with the tallying authorities

Practical Everlasting Privacy: Everlasting privacy towards the public
[ACKR13]

External adversary has the same view with the voters

Excludes insiders

Anonymous casting (+ blind signatures) [FOO92]

- Disassociate identity with ballot
- Provide a blind signature to identified ballot to signal eligibility
- Identity is no longer required
- Blinded ballots cast anonymously

A further advantage: TA is not required to be trusted for privacy

- Pedersen commitment $\text{Commit}(m, r) = g^m h^r$ $g, h \leftarrow_{\$} \mathbb{G}$
- Homomorphic:
 $\text{Commit}(m_1, r_1) \cdot \text{Commit}(m_2, r_2) = \text{Commit}(m_1 + m_2, r_1 + r_2)$
- Perfectly hiding - computationally binding
- Problem: ElGamal cannot be used for encrypting m, r
- Use a compatible cryptosystem (many approaches)

Vote(i, \mathbf{vt}_i)

$(\mathbf{b}_i, \text{ok}_i) \leftarrow \text{Commit}(\text{ck}, \mathbf{vt}_i)$

$[\mathbf{vt}_i] \leftarrow \text{Enc}(\text{pk}_{\text{TA}}, \mathbf{vt}_i)$

$[\text{ok}_i] \leftarrow \text{Enc}(\text{pk}_{\text{TA}}, \text{ok}_i)$

$\text{BB} \leftarrow \mathbf{b}_i$

Send $[\mathbf{vt}_i], [\text{ok}_i]$ to TA

Tally(sk_{TA})

$\mathbf{b} = \prod_{i=1}^n \mathbf{b}_i$

$[\mathbf{vt}] = \text{Enc}(\text{pk}_{\text{TA}}, \sum_{i=1}^n \mathbf{vt}_i) = \prod_{i=1}^n [\mathbf{vt}_i]$

$\mathbf{vt} = \text{Dec}(\text{sk}_{\text{TA}}, [\mathbf{vt}])$

$[\text{ok}] = \text{Enc}(\text{pk}_{\text{TA}}, \sum_{i=1}^n \text{ok}_i) = \prod_{i=1}^n [\text{ok}_i]$

$\text{ok} = \text{Dec}(\text{sk}_{\text{TA}}, [\text{ok}])$

if CS.Open($\mathbf{b}, \mathbf{vt}, \text{ok}$) = 1 **then**

return \mathbf{vt}

endif

Helios Extensions for receipt-freeness and participation privacy

Intuition

Privacy does not deal with malicious *voters*, that willingly interact with an attacker before/after voting to sell their votes

Assumption: \mathcal{A} does not monitor voter during vote casting. Only at distinct moments before / after.

Receipt-Freeness

A (malicious) voter cannot prove how she voted, even if she wants to

Helios is not receipt-free

The malicious voter can reveal the randomness used in ballot creation (encryption) \mathcal{A} can recompute the encryption and check the BB for the ballot

Revoting does not help. \mathcal{A} can check which ballot finally appears in the published BB.

- The voter computes and signs the ballot (\mathbf{b}, σ)
- The EA converts reencrypts (rerandomizes) \mathbf{b} to \mathbf{b}'
- The voter cannot longer prove how they voted
- The EA proves that \mathbf{b}, \mathbf{b}' contain the same voter
- The proof must be non-transferrable
- Designated - Verifier (Reencryption) Proofs
 - Includes the public key of the verifier
 - The proof ascertains that:
 - **The statement is valid** OR **I know the private key of the verifier**
- The voter signs \mathbf{b}'

How to remove interactivity?

Non-Interactive Solutions

BeleniosRF [CCFG16]

- Rerandomization Server.
- Changes the randomization contained in the ballot.
- The voter is not alone in contributing randomness.
- So the voter has no use in revealing it.
- The signatures must adapt.
- Collusion of EA and rerandomisation server could change vote.

Helios-KTV [KTV15]

- Deniable vote updating
- The voter casts \mathbf{b}_1 (the solded vote $\mathbf{v}\mathbf{t}_1$)
- Later updates the ballot to \mathbf{b}'_1 (a new vote $\mathbf{v}\mathbf{t}'_1$)
- Anonymization layer required
- Revoting might not be allowed
- The adversarial ballot should be cast first or the voter should know which option will be for sale

Both define receipt-freeness as an extension of BPRIV

Signatures on Randomizable Ciphertexts i

- $(dk, pk, vk, sk) \leftarrow \text{KGen}(1^\lambda)$
- $\sigma \leftarrow \text{Sign}(sk, \mathbf{m}, pk)$
- $\{0, 1\} \leftarrow \text{Verify}(vk, \sigma, \mathbf{m})$
- $c \leftarrow \text{Enc}(pk, \mathbf{m}, vk)$
- $\mathbf{m} \leftarrow \text{Dec}(dk, c)$
- $(c', \sigma') \leftarrow \text{Random}(pk, vk, c, r', \sigma, s')$
- $\text{Vote}(id, pk, vk, \mathbf{vt}) = (c = \text{Enc}(pk, \mathbf{m}, vk), \text{Sign}(sk, c, pk))$
- $\text{Append}(\mathbf{b}) = \text{Random}(pk, vk, \mathbf{b})$

Verifiability: Validate the signatures along with the proofs

\mathcal{A} has additionally access to the following oracles:

- **Register**(i): Generate credentials for the voter
- **Corrupt**(i): Corrupt a voter for receipt-freeness (get the voter's key)
- **Receipt**($\mathbf{b}_0, \mathbf{b}_1$): The corrupted voter casts two ballots to the respective BB. They are assumed to encode information that will allow vote selling (e.g. set last bit to one)

\mathcal{A} must guess which BB is tallied

A ballot \mathbf{b}_b submitted through **Receipt**($\mathbf{b}_0, \mathbf{b}_1$) can be swapped with \mathbf{b}_{1-b} without \mathcal{A} noticing

BPRIV for receipt-freeness ii

Algorithm 3: BPRIV extensions for BeleniosRF

Oracle Register(i)

```
(pki, ski) ← Register( $i$ )
VE1 ← ( $i$ , pki, ski)
return pki
```

Oracle Corrupt(i)

```
if  $i \in V_{E1}$  then
  VCorr ← ( $i$ , pki, ski)
  return (pki, ski)
```

Oracle Receipt(i , \mathbf{b}_0 , \mathbf{b}_1)

```
if  $i \in V_{Corr}$  then
  if
    Valid( $\mathbf{b}_0$ , BB0) AND Valid( $\mathbf{b}_1$ , BB1)
  then
    BB0 ←  $\mathbf{b}_0$ 
    BB1 ←  $\mathbf{b}_1$ 
```

- \mathcal{A} acts like a voter
- The ballot \mathbf{b}_0 can be replaced with \mathbf{b}_1 (by the rerandomization server) without \mathcal{A} noticing
- No strategy on behalf of the voter needs to be applied
 - More practical schemes (single pass)
 - Revoting is not always allowed
- Helios is not receipt-free: \mathcal{A} can encode information in BB₀, BB₁ (randomness) to distinguish them

Participation privacy: The voter cannot prove *if she voted or abstained*.

More serious attack for Internet voting due to larger scale

Basic idea: *dummy* ballots for all registered voters

- cast by a TTP - the posting trustee
 - at random voting intervals during the voting phase
 - contain a null vote
 - many dummy ballots per voter - id
- $$\mathbf{VoteDummy}(i) = \mathbf{Vote}(i, pk, \cdot, 0)$$
- before tallying: homomorphically aggregate all ballots per voter id
 - the end result contains the real vote of voter id

For Receipt-Freeness: Deniable vote updating

- the \mathcal{A} wants to cast a ballot $\mathbf{b}_{\mathcal{A}} = \mathbf{Vote}(i, pk, sk_i, \mathbf{vt}_{\mathcal{A}})$
- the voter obeys
- at a different time (earlier or later) the voter casts a different ballot \mathbf{b}_v that updates to correspond to vote \mathbf{vt}_v
- **DeniablyUpdate** $(i, \mathbf{vt}_v, \mathbf{vt}_{\mathcal{A}}) = \mathbf{Vote}(i, pk, sk_i, \mathbf{vt}_v - \mathbf{vt}_{\mathcal{A}})$
- need to know previous vote

Assumptions

- available public key infrastructure for voter credentials
- anonymous channel
- \mathcal{A} does not watch voter during the complete election
- trusted posting trustee, BB, TA

Extend BPRIV using two oracles

Participation Privacy

Main idea: Vote swapping with abstention.

\mathcal{A} cannot distinguish if V_{i_0} abstained and V_{i_1} voted at most k times or vice versa by only viewing BB_b

Algorithm 4: BPRIV extension for KTV-Helios (Participation Privacy)

Oracle $\text{VoteAbstain}_{\mathbb{P}}(i_0, i_1, \mathbf{vt}_1, \dots, \mathbf{vt}_k)$

$n_b, n_{1-b} \leftarrow \$_{\mathbb{P}}$

$\{\mathbf{b}_{i_b, i}\}_{i=1}^{n_b} \leftarrow \text{VoteDummy}(i_b)$

$\{\mathbf{b}_{i_{1-b}, i}\}_{i=1}^{n_{1-b}} \leftarrow \text{VoteDummy}(i_{1-b})$

$\{\mathbf{b}_{i_b, i} \leftarrow \text{Vote}((i_b, i_0), \text{pk}, \text{sk}_{i_b}, \mathbf{vt}_i)\}_{i=1}^k$

$\{\mathbf{b}_{i_b, i} \leftarrow \text{Vote}((i_b, i_1), \text{pk}, \text{sk}_{i_b}, \mathbf{vt}_i)\}_{i=1}^k$

$\{BB_b \leftarrow \mathbf{b}_{i_b, i}\}_{i=1}^{k+n_b}$

$\{BB_{1-b} \leftarrow \mathbf{b}_{i_{1-b}, i}\}_{i=1}^{k+n_{1-b}}$

For $b = 0$ the voter sells her vote.

For $b = 1$ the voter tries to fool the coercer and executes

DeniablyUpdate

distinguishing factor - number of ballots in BB_0, BB_1 - obfuscation using **VoteDummy**

Algorithm 5: BPRIV extensions for KTV-Helios

Oracle **Receipt**($i, \mathbf{vt}_{\mathcal{A}}, \mathbf{vt}_v$)

$\mathbf{b}_{\mathcal{A}} = \mathbf{Vote}(i, \text{pk}, \text{sk}_i, \mathbf{vt}_{\mathcal{A}})$

$BB_0 \leftarrow \mathbf{b}_{\mathcal{A}}$

$BB_1 \leftarrow \mathbf{b}_{\mathcal{A}}$

$\mathbf{b}_v = \mathbf{DeniablyUpdate}(i, \mathbf{vt}_v, \mathbf{vt}_{\mathcal{A}})$

$BB_1 \leftarrow \mathbf{b}_v$

$BB_0 \leftarrow \mathbf{VoteDummy}(i)$

Coercion Resistance

A stronger adversary

Active methods for attack:

- Vote for a specific candidate / randomly
- Totally abstain from voting
- Yield private keys - allow to be simulated

Passive methods for attack:

- Monitor voting system throughout election
- Same for voter except for a moment of privacy

Goal: Internet voting

Note: Coercion Resistance \Rightarrow Receipt-Freeness

The JCJ coercion resistance framework [JCJ05]

Intuition

\mathcal{A} will not be motivated to attack, if he cannot check if the attack succeeds

Techniques

- Multiple votes per voter
- Authentication using anonymous credentials

Registration phase: voter registers a real credential

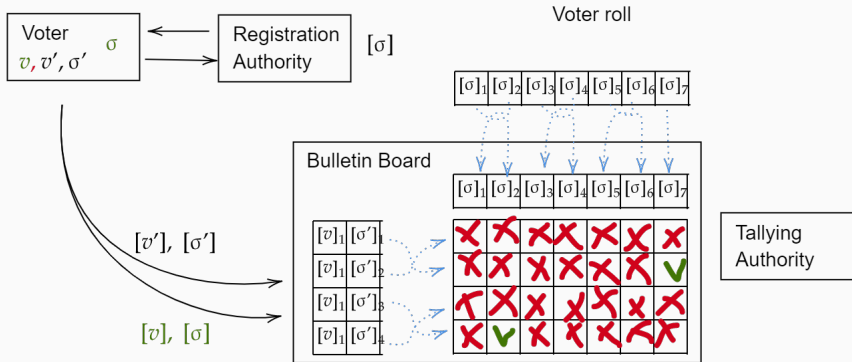
During coercion attack: voter supplies a fake credential (indistinguishable)

During moment of privacy: voter casts the real vote

During tallying: the TA must filter out **fake and duplicate votes** in a *verifiable manner* without disclosing which votes are not counted

How: Blind comparisons in the backend against a voter roll

The scheme



A key component: $\text{PET}(\text{Enc}_{\text{pk}}(m_1), \text{Enc}_{\text{pk}}(m_2)) = 1 \Leftrightarrow m_1 = m_2$

Plaintext Equivalence Test

Algorithm 6: PET for ElGamal ciphertexts

Input : $\mathbb{G}, g, q, \text{pk}_i: \prod_i^t \text{pk}_i = \text{pk}, c = (c_1, c_2), c' = (c'_1, c'_2)$

Private Input: $\text{sk}_i \in \mathbb{Z}_q: \sum_i^t \text{sk}_i = \text{sk}$

Output : $\{0, 1\}$

$$c_{\text{PET}} := \frac{c}{c'} = \left(\frac{c_1}{c'_1}, \frac{c_2}{c'_2} \right)$$

$$z_i \leftarrow \$_{\mathbb{Z}_q}$$

$$c_{i,\text{PET}} := c_{\text{PET}}^{z_i} = (c_{i1}, c_{i2}) = \left(\left(\frac{c_1}{c'_1} \right)^{z_i}, \left(\frac{c_2}{c'_2} \right)^{z_i} \right)$$

$$\pi_{i1} \leftarrow \text{NIZK} \left\{ \left(\mathbb{G}, g, q, \text{pk}, c_{\text{PET}}, c_{i,\text{PET}} \right), (z_i) : c_{i,\text{PET}} = c_{\text{PET}}^{z_i} \right\}$$

Publish $(c_{i,\text{PET}}, \pi_{i1})$ and wait until all players have posted. Verify the proofs π_{i1} posted from other players

$$\phi := \prod_i^t c_{i,\text{PET}} = \left(\prod_i c_{i1}, \prod_i c_{i2} \right) = \left(c_{1i}^{\sum_i z_i}, c_{2i}^{\sum_i z_i} \right) = (x, y)$$

$$\psi_i := x^{\text{sk}_i}$$

$$\pi_{i2} \leftarrow \text{NIZK} \left\{ \left(\mathbb{G}, g, q, \text{pk}, \psi_i \right), (\text{sk}_i) : \psi_i = x^{\text{sk}_i} \right\}$$

Publish (ψ_i, π_{i2}) wait until all players have posted. Verify the proofs π_{i2} posted from other players

$$\rho := y / \prod_i^t \psi_i$$

$$\text{return } \rho = 1$$

Note: The strong Fiat-Shamir heuristic must be used If not verifiability can be broken

Assumptions

- Moment Of Privacy
- Untappable Registration (occurs one / used in multiple elections)
 - Secure transcript erasure
 - Simulation of transcript
- Anonymous casting
 - For forced abstention attack
- Coercer uncertainty about voter behavior
 - If all vote, then the abstention attack will always succeed
 - The voting authorities may inject chaffe votes on purpose

Algorithm 7: Real Coercion resistance game

```
(prms,  $V_{EL}, CS$ )  $\leftarrow$  Setup( $1^\lambda$ )
 $\{(sk_i, pk_i) \leftarrow \text{Register}(sk_{RA}, i)\}_{i=1}^n$ 
 $V_{CORR} \leftarrow \mathcal{A}(\text{corrupt})$  // Adversary corrupts voters
 $(j, vt_j) \leftarrow \mathcal{A}(V_{HON}, \text{coerce})$  // Adversary coerces voter j
 $b \leftarrow \mathcal{S}\{0, 1\}$ 
if  $b = 0$  then
    |  $sk_j^* \leftarrow \text{fakekey}(j)$  // generate fake credential
    |  $b_j \leftarrow \text{Vote}(j, vt_j, sk_j)$  // Moment of privacy
else
    |  $sk_j^* \leftarrow sk_j$  // yield credential
end
 $\{b_i \leftarrow \text{Vote}(i, vt_i, sk_i), \}_{i=1}^{|V_{HON}|, \mathcal{D}}$ 
 $BB \leftarrow \text{Vote}(j, vt_j, sk_j^*) \{BB \leftarrow \mathcal{A}(sk_i, vt_i, \text{cast})\}_{i=1}^{|V_{CORR}|}$ 
 $(T, \pi_T, \Gamma) := \text{Tally}(BB, sk_{TA})$ 
 $b' \leftarrow \mathcal{A}(T, \pi_T, \Gamma, BB, \text{guess})$ 
return  $b = b'$ 
```

Algorithm 8: Ideal Coercion resistance game

```
(prms,  $V_{EL}, CS$ )  $\leftarrow$  Setup( $1^\lambda$ )
 $\{(sk_i, pk_i) \leftarrow \text{Register}(sk_{RA}, i)\}_{i=1}^n$ 
 $V_{\text{Corr}} \leftarrow \mathcal{A}(\text{corrupt})$  // Adversary corrupts voters
 $(j, vt_j) \leftarrow \mathcal{A}(V_{\text{Hon}}, \text{coerce})$  // Adversary coerces voter j
 $b \leftarrow \mathcal{S} \{0, 1\}$ 
if  $b = 0$  then
  |  $b_j \leftarrow \text{Vote}(j, vt_j, sk_j)$ 
end
// Moment of privacy
 $sk_j^* \leftarrow sk_j$  // always yield credential
 $\{b_i \leftarrow \text{Vote}(i, vt_i, sk_i)\}_{i=1}^{|V_{\text{Hon}}|, \mathcal{D}}$ 
 $BB \leftarrow \text{Vote}(j, vt_j, sk_j^*)$ 
 $\{BB \leftarrow \mathcal{A}(sk_i, vt_i, \text{cast})\}_{i=1}^{|V_{\text{Corr}}|}$ 
 $T := \text{ideal\_tally}(BB, sk_{TA})$ 
 $b' \leftarrow \mathcal{A}(T, \text{guess})$ 
return  $b = b'$ 
```

Why the ideal experiment?

- An alternative: Distinguish between $b = 0$ and $b = 1$?
- The tally might help the coercer distinguish if the coercion attempt succeeded
- For instance: The voter instruct a vote for 'Alice' but no 'Alice' votes are found (regardless of the cryptographic primitives used)
- We need to measure the effect of the cryptographic primitives

Ideal tallying functionality

- Ballots cast by V_{Hon} are treated normally
- Ballots cast by \mathcal{A} are added to the result
- Also performs weeding of double votes based on the extracted credential
- If $b = 0$ votes with sk_j^* are *not* counted
- If $b = 1$ votes with sk_j^* are counted

Main drawback Quadratic tallying complexity: $\mathcal{O}(v^2) + \mathcal{O}(nv)$ for duplicate detection and tallying

Goal $\mathcal{O}(n + v)$

3 approaches for better efficiency:

Anonymity sets

The ballot contains

- The current credential
- The real credential from the voter roll (rerandomized)
- Some $\beta - 1$ other random credentials from the BB
- The PET takes place only among the credentials of the ballot

Blinded hashing

- The credential $[\sigma]$ is blinded to obtain $[\sigma]^z$
- The value z is jointly computed by the RA members
- Instead of PET, $[\sigma]^z$ is decrypted to obtain σ^z (credential fingerprint)
- This fingerprint is sent through a hashtable $\{(\sigma^z, \mathbf{b})\}$
- For verifiability: σ^z is made public so that everybody can check the hashtable

Vulnerable to the tagging attack (Pfitzmann)

- \mathcal{A} forces a voter to reveal the credential
- Then \mathcal{A} posts two ballots one with $[\sigma]$ and one with $[\sigma^2]$
- Two fingerprints will be produced σ^z and one with σ^{2z}
- \mathcal{A} squares all elements in the BB.
- If a square matches another element and both have been discarded then \mathcal{A} learns that σ was fake

Note: This attack applies only to fake credentials, not duplicates

So $\mathcal{O}(v^2)$ becomes $\mathcal{O}(v)$

Structured credentials

Check of validity is self contained in the credential

- A credential is a tuple $(r \leftarrow \mathbb{Z}_q, A \leftarrow \mathbb{G}, B = A^y, C = A^{x+rx y})$ where $x, y \leftarrow \mathbb{Z}_q$ are secret keys of the RA
- r should be kept secret by the voter
- A fake credential can be made by selecting a new r
- If (r, A, B, C) is valid then (r, A^l, B^l, C^l) is valid
- $\mathbf{b} = ([\mathbf{vt}], [A], [A^r], [B^r], [C], O^r) = (\cdot, U, V, W, Z, \cdot)$
- Duplicates are identified by o^r
- Ballot validity: if $Z \cdot U^{-x} \cdot W^{-x}$ decrypts to 1

Decentralized voting

2 Voting paradigms

- Large scale elections
 - Involvement of authorities (mixing / tallying)
 - Trust required for some properties
 - Each voter is only interested to cast their ballot (vote & go)
 - Existence of BB: contains all voting public data (broadcast channel with memory)
- Small(er) scale elections (boardroom)
 - Conducted by the voters themselves
 - No entity plays a special part
 - Robustness is more important: A voter cannot disrupt the election
 - Private channels lead to disputes

Can these two paradigms be combined?

- Self-Tallying (open tallying)
Any (external) entity should be able to count the ballots
(Implies verifiability)
- Perfect Ballot Secrecy
The partial election result for a particular subset of voters can be obtained only by a coalition that contains the rest of the voters
- Dispute-Freeness Embedded mechanisms avert disputes and make the participants follow the protocol (accountability)

Can these properties be achieved while minimizing communication complexity and voter-to-voter interaction?

Relationships

Theorem

Self-tallying is incompatible with robustness and privacy (at the same time)

Assume: Self-tallying and robust

Assume: $n - 1$ out of n voters show up

By self tallying: Anyone can compute the result

But: The same computation can take place even if all voters show up

This reveals the preference of the last (any) voter

Theorem

Robustness through threshold secret sharing is incompatible with perfect ballot secrecy

For perfect ballot secrecy the threshold must be set to n

But this is not robust

A solution: Decentralized voting with a BB (can be replaced with a blockchain)

Anonymous Voting by 2-Round Public Discussion - [HRZ10]

Preparation

Select a group $\mathbb{G} = \langle g \rangle$

Each voter has an identity V_i

Selects $a_i \leftarrow \mathbb{Z}_q$

Round 1 - Commitment

Each V_i posts g^{a_i} , $\text{NIZK}\{(a_i), g^{a_i}\}$

When this phase ends compute:

$$\prod_{j=1}^{i-1} g^{a_j} / \prod_{j=i+1}^n g^{a_j} = g^{y_i}$$

for some *unknown* $y_i \in \mathbb{Z}_q$

Round 2 - Voting

Each V_i selects $v_i \in \{0, 1\}$ and posts $(g^{y_i})^{a_i} g^{v_i}$

Round 3 - Self-tallying

Everyone computes

$$\prod_{i=1}^n (g^{y_i})^{a_i} g^{v_i} = \prod_{i=1}^n g^{v_i} = g^{\sum_{i=1}^n v_i}$$

Solve a simple DLP

Correctness

$$\sum_{i=1}^n a_i y_i = \sum_{i=1}^n \sum_{j<i} a_i a_j - \sum_{i=1}^n \sum_{j>i} a_i a_j = 0 \text{ since: } y_i = \sum_{j<i} a_j - \sum_{j>i} a_j$$

	x_1	x_2	x_3	x_4	x_5
x_1		-	-	-	-
x_2	+		-	-	-
x_3	+	+		-	-
x_4	+	+	+		-
x_5	+	+	+	+	

Problems

Robustness: If someone refuses to vote then the result cannot be computed

Fairness: The last voter can learn the result

Improvements for robustness and fairness [KSRH12]

Robustness

Assume that only voters in \mathcal{L} have voted.

Recovery Round

Each $V_i \in \mathcal{L}$ compute:

$$\hat{h}_i = \frac{\prod_{j \in \{i+1, n\} \setminus \mathcal{L}} g^{a_j}}{\prod_{j \in \{1, i-1\} \setminus \mathcal{L}} g^{a_j}}$$

and posts cancellation tokens $\hat{h}_i^{a_i}$, $\text{NIZK}\{(a_i) : \log_g g^{a_i} = \log_{\hat{h}} \hat{h}^{a_i}\}$

Tallying becomes: $V = \prod_{i=1}^n b_i = \prod_{i \in \mathcal{L}} \hat{h}_i^{a_i} (g^{y_i})^{a_i} g^{v_i} = g^{\sum_{i=1}^n v_i}$

No	First round	Second round	Third round	Recovery
1	g^{x_1}	commitment	$g^{x_1 y_1} = g^{x_1(-x_2-x_3-x_4-x_5)}$	$\hat{h}_1^{x_1} = g^{x_1(x_2+x_4)}$
2	g^{x_2}	commitment	Abort	-
3	g^{x_3}	commitment	$g^{x_3 y_3} = g^{x_3(x_1+x_2-x_4-x_5)}$	$\hat{h}_3^{x_3} = g^{x_3(x_4-x_2)}$
4	g^{x_4}	commitment	Abort	-
5	g^{x_5}	commitment	$g^{x_5 y_5} = g^{x_5(x_1+x_2+x_3+x_4)}$	$\hat{h}_5^{x_5} = g^{x_5(-x_2-x_4)}$

- Implementation of [HRZ10] using Ethereum
- Smart contracts (voting, registration, tallying)
- Voters are authenticated with their Ethereum user-controlled accounts
- Ethereum restrictions:
 - integers of 256 bits
 - expensive cryptographic computations
 - one vote or six registrations per block
 - small number of allowed local variables
 - order of transactions in a block and timers
- Maximum number of voters: 50 (due to gas limit)
 - Linear number of operations for Tally and Voter List
- Cost per voter: 0.73\$

Improvements

- Organize voters in Merkle Tree only the root is stored (256 bits)
- Instead of voter list a voter provides a proof of membership
- Tally off-chain by an **untrusted** tallier
 - Publish computation trace in Merkle Tree $((i, t_i))$
 - Subject to verification

```
def TallyVotes(c: array []):  
    t = 1  
    for i=1 to n:  
        t = Mul(c[i], t)  
    return t
```

Step i	c_i	t_{i-1}	t_i
1	c_1	$t_0 = 1$	$t_1 = c_1$
2	c_2	t_1	$t_2 = c_2 \cdot t_1$
.
n	c_n	t_{n-1}	$t_n = c_n \cdot t_{n-1}$

Blockchain and voting i

Conceptual similarity between blockchain and the BB

- Append-only
- Broadcast channel
- No central authority - anyone can be a miner (given enough computing power)
- Pseudonymity

Good for universal/individual verifiability (recorded as cast)

But...

- Registration/authentication/eligibility verifiability are inherently centralized
- Does not help with verifying voter intent
- Does not help with coercion-resistance / receipt-freeness

- Intensifies threats associated with everlasting privacy
- Is it actually decentralized? (concentration of mining power)

To sum up... 'using Blockchain for voting solves a *small* part of the problem with an unnecessarily big hammer' (Ben Adida, 2017)

However... it might be useful for different types of elections - new election paradigms on a smaller scale with many different permission-types of blockchains

Relations between properties

One can have IV without UV

- Construct a scheme with $(\mathbf{vt}_i, r_i) \leftarrow \mathbf{Vote}(i, \mathbf{vt}_i)$
- Individual verifiability because of r_i
- Allows ballots stuffing

One can have UV without IV if everyone votes

- Construct a scheme with $\mathbf{vt}_i \leftarrow \mathbf{Vote}(i, \mathbf{vt}_i)$
- Everybody can calculate the same tally
- Clash for same preferences

EV implies IV

- Nobody can construct a ballot unless they know sk_i (private voter credential)
- sk_i allows individual verifiability

Universal verifiability is incompatible with unconditional privacy

- Unconstrained \mathcal{A}
- Compute election tally of a sublist of $n - 1$ voters
- Decryption and comparison with the tally of n voters
- Reveal the option of the remaining voter

Privacy implies individual verifiability

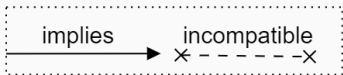
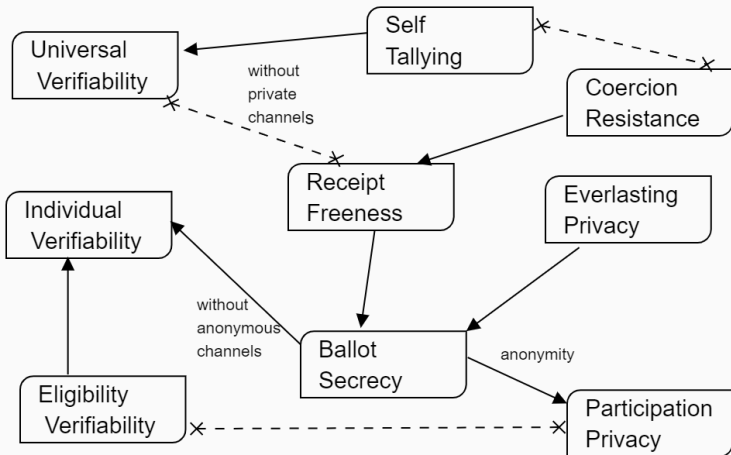
- Assume no individual verifiability
- A corrupt EA can replace all ballots except one
- Learn the particular preference

Receipt-freeness is incompatible with universal/individual verifiability

- Encryption randomness can be used as a receipt
- Without randomness: receipt-freeness
- With randomness: verifiability
- No private or anonymous channels are used

- Receipt freeness implies privacy
- Everlasting privacy implies privacy
- The reverse does not hold
- Everlasting privacy is related to coercion resistance
- Coercion resistance and receipt-freeness: Depends on the model




Overview of relationships








Open questions




- Security Analysis of Zeus
 - Check if the pitfalls of the Fiat - Shamir Heuristic apply
 - Formal proof of verifiability and privacy
 - Security issues
- Decentralized voting with coercion resistance
 - Ring signatures
 - Self-tallying incompatibility
- Relations between properties
 - Formal proofs

References

-  Myrto Arapinis, Véronique Cortier, Steve Kremer, and Mark Ryan, *Practical everlasting privacy*, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 7796 LNCS, 2013, pp. 21–40.
-  David Bernhard, Véronique Cortier, David Galindo, Olivier Pereira, and Bogdan Warinschi, *Sok: A comprehensive analysis of game-based ballot privacy definitions*, 2015 IEEE Symposium on Security and Privacy, 2015, pp. 499–516.
-  Mihir Bellare and Amit Sahai, *Non-malleable encryption: Equivalence between two notions, and an indistinguishability-based characterization*, Advances in Cryptology — CRYPTO' 99 (Michael Wiener, ed.).

-  Pyrros Chaidos, Véronique Cortier, Georg Fuchsbauer, and David Galindo, *Beleniosrf: A non-interactive receipt-free electronic voting scheme*, Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016, pp. 1614–1625.
-  Véronique Cortier and Ben Smyth, *Attacking and fixing helios: An analysis of ballot secrecy*, J. Comput. Secur. (2013), 89–148.
-  Denise Demirel, J Van De Graaf, and R Araújo, *Improving Helios with Everlasting Privacy Towards the Public*, EVT/WOTE'12 Proceedings of the 2012 international conference on Electronic Voting Technology/Workshop on Trustworthy Elections (2012).
-  Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta, *A practical secret voting scheme for large scale elections*, AUSCRYPT (Jennifer Seberry and Yuliang Zheng, eds.), LNCS, Springer, 1992, pp. 244–251.

-  Feng Hao, Peter Ryan, and Piotr Zielinski, *Anonymous voting by two-round public discussion*, Information Security, IET **4** (2010), 62 – 67.
-  Martin Hirt and Kazue Sako, *Efficient receipt-free voting based on homomorphic encryption*, EUROCRYPT 2000.
-  Ari Juels, Dario Catalano, and Markus Jakobsson, *Coercion-resistant electronic elections*, Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society, WPES 2005, Alexandria, VA, USA, November 7, 2005, ACM, 2005, pp. 61–70.
-  Oksana Kulyk, Vanessa Teague, and Melanie Volkamer, *Extending helios towards private eligibility verifiability*, VoteID, Lecture Notes in Computer Science, vol. 9269, Springer, 2015, pp. 57–73.
-  Aggelos Kiayias and Moti Yung, *Self-tallying elections and perfect ballot secrecy*, Public Key Cryptography, 2002, pp. 141–158.

-  Tal Moran and Moni Naor, *Receipt-Free Universally-Verifiable Voting with Everlasting Privacy*, 2006, pp. 373–392.
-  Patrick McCorry, Siamak Shahandashti, and Feng Hao, *A smart contract for boardroom voting with maximum voter privacy*, pp. 357–375, 01 2017.
-  Mohamed Seifelnasr and Hisham Galal, *Scalable open-vote network on ethereum*, pp. 436–450, 08 2020.