Private Voting

Dan Boneh
Stanford University
Privacy questions in DAOs

- Private voting
- Private delegation
- Private treasuries
- Private membership – legal implications?? (Jeff)
Many voting mechanisms

- One token one vote
- Quadratic voting
- Badge voting

... all of these need privacy to get at voters’ true intent!
Why private voting ??
Nouners many times aren’t necessarily voting for what they believe is best. Instead, they feel trapped into *quid pro quo* voting, afraid that their vote could reflect poorly on their image, and/or affect the likelihood of getting their own prop through when the time comes. Conversely, it occurs that Nouners vote in favor or against a proposal *based on how the proposer voted for their past props.*
Let’s vote on it (not privately):

**Description**
- 3 months, 3 zero-knowledge teams, funded by Nouns to design open source solutions for private voting.
- All outputs are open source, and include detailed designs and proofs of concept.
- 70K USDC per team + 20K USDC marketing budget to appear on zk pods and newsletters. Total: 230K USDC.

Funds are managed in a [3/5 Gnosis Safe](https://gnosis-safe.io) managed by Elad, David, Solimander, Vapeape and Will.

**Mandated Round: Private Voting Research Sprint**

Proposed by delegate.el4d.eth at 0x7b1ba

Switch to delegate view

<table>
<thead>
<tr>
<th>For</th>
<th>166</th>
</tr>
</thead>
<tbody>
<tr>
<td>Against</td>
<td>75</td>
</tr>
<tr>
<td>Abstain</td>
<td>19</td>
</tr>
</tbody>
</table>

Jan. 31, 2023
19 submissions! Three selected by a vote

Team 1

👑 DeFROST: Poseidon 🫙
We have designed a custom cryptographic scheme we call DeFROST to satisfy all Nouns DAO private voting requirements.

Team 2

👑 Aragon and Aztec join forces to bring private voting to...
State-of-the-art DAO private voting with minimal off-chain dependency via Ethereum storage proofs & timelapse encryption

Team 3

👑 Nouns Vortex
A privacy preserving on-chain voting for Nouns DAO; the proposal by team Mizu
The Time NounsDAO Got Private Voting

Aztec x Aragon bring private on-chain voting to Nouns

Aztec Labs · Follow
Published in Aztec · 4 min read · Mar 8

https://medium.com/aztec-protocol/the-time-nounsdao-got-private-voting-4336fe4a2c29
How to vote privately?

High level goals:

• Only authorized voters can vote, and vote only once
• A voter cannot be linked to their vote
• End-to-end verifiability:
  voters can check that votes are counted as cast
• Voter cannot prove how they voted (no vote selling / coercion)
Three core techniques

(1) Mix nets

(2) Homomorphic encryption

(3) Blind signatures

**Warning:** we only describe the core ideas.
- Real-world secure voting is much more complicated

see, e.g., VotingWorks and ElectionGuard
Method 1: Mix nets

First: Public Key Encryption (PKE):

\[ \text{KeyGen()} \rightarrow (pk, sk); \quad E(pk, m) \rightarrow ct; \quad D(sk, ct) \rightarrow m \]
Method 1: Mix nets

\[ \text{KeyGen()} \rightarrow (pk, sk); \quad E(pk, m) \rightarrow ct; \quad D(sk, ct) \rightarrow m \]

Re-randomizable public key encryption: \[ \text{ReRand}(pk, ct) \rightarrow ct' \]

For all \( m_0, m_1 \): if \( ct_0 \leftarrow E(pk, m_0) \) and \( ct_1 \leftarrow E(pk, m_1) \) then

\( (ct_0, ct_1, \text{ReRand}(pk, ct_0)) \) “looks the same as” \( (ct_0, ct_1, \text{ReRand}(pk, ct_1)) \)
Method 1: Mix nets

KeyGen() → (pk, sk);  E(pk, m) → ct;  D(sk, ct) → m

Threshold decryption: ≥t parties are needed to decrypt ct

- KeyGen(n, t) → pk, sk₁, sk₂, ..., skₙ

\[ E(pk, m) \rightarrow ct \]

\[ sk₁ \rightarrow D(sk₁, ct) \rightarrow m₁ \]

\[ sk₂ \rightarrow D(sk₂, ct) \rightarrow m₂ \]

\[ sk₃ \rightarrow D(sk₃, ct) \rightarrow m₃ \]

Rest of parties compute shares m₁, m₂, m₃.

Combiner produces decrypted m.

Decryption share threshold=2
Warmup: private voting using a single mix node

Mix node

E(pk, vote1)

E(pk, vote2)

E(pk, vote3)

(1) choose a random permutation
(2) re-randomize all ciphertexts
(3) output in permuted order

trustees

permuted votes

threshold decrypt

(sk_1, sk_2, ..., sk_n)

Is this private? Does the election have integrity?
Warmup: private voting using a single mix node

Trustees are trusted to only decrypt output of mix (not input)
(an adversary cannot get $t$ trustees to decrypt the input)

Mix node only accepts votes from authorized voters (one per voter)

Problems:

1. The mix node will know how everyone voted
2. The mix node can modify the output ciphertexts
Fixing problem 1: multiple mix nodes

Privacy for final votes if one honest mix node deletes its permutation (and mixes do not modify the votes)
Fixing problem 2: proof of shuffle

Mix #1

Mix #2

Mix #3

\[ ct_1 \]

\[ ct_2 \]

\[ ct_3 \]

\[ \pi_1 \]

\[ \pi_2 \]

\[ \pi_3 \]

\[ ct'_3 \]

\[ ct'_2 \]

\[ ct'_1 \]

Trustees

permuted votes

ZK proofs that each output is a permutation of the input

anyone can check proofs
What is this ZK?  (very briefly)

Proof that a statement is true without revealing why

Example: prover wants to prove that a Soduko puzzle is solvable

Victor learns witness exists, but learns nothing else (formally: proof can be simulated without the witness)

Prover Peggy

Verifier Victor

can be made non-interactive
What is this ZK? (very briefly)

Proof that a statement is true without revealing why

Example: prover wants to prove that a Soduko puzzle is solvable

Prover Peggy  
Verifer Victor

Victor learns witness exists, but learns nothing else  
(formally: proof can be simulated without the witness)

Proof $\pi$  
can be made non-interactive
ZK proof of shuffle (an example)

**Prover Peggy**

**Verifier Victor**

**Witness:**

- Permutation $\sigma$
- $\sigma(\text{inVotes}) = \text{outVotes}$
- and randomizers

**Statement**

- $ct_1$
- $ct_2$
- $ct_3$

- $ct'_1$
- $ct'_2$
- $ct'_3$

**Rand. $\rho$**

- $t \leftarrow \text{ReRand}(\rho(\text{inp}))$

- $b \in \{0,1\}$: reveal (perm: inp to $t$) or (perm: out to $t$)

- $\tau \leftarrow \rho$ or $\tau \leftarrow \rho\sigma^{-1}$ (and randomizers)
ZK proof of shuffle (an example)

Prover Peggy

Verifier Victor

witness:
permutation $\sigma$
$\sigma(\text{inVotes}) = \text{outVotes}$
and randomizers

Why is this sound?
Why is it zero knowledge?
Final decrypted votes are not linked to voters,
Anyone can check that votes were not changed

For a DAO vote: store these on the blockchain
Method 2: Homomorphic encryption

\[
\text{KeyGen}() \rightarrow (pk, sk); \quad \text{E}(pk, m) \rightarrow ct; \quad \text{D}(sk, ct) \rightarrow m
\]

Additive homomorphic encryption: \( \text{msg space} \subseteq \mathbb{F} \)

Let \( ct_1, ct_2 \) satisfy \( \text{D}(sk, ct_1) = m_1 \) and \( \text{D}(sk, ct_2) = m_2 \)

\[
\text{AddCT}(pk, ct_1, ct_2) \rightarrow ct
\]

where \( ct \) is a fresh encryption of \( m_1 + m_2 \)
Method 2: Homomorphic encryption

\[(v_a \in \{0,1\}, w_a) \quad (v_b, w_b) \quad (v_c, w_c)\]

\[E(pk, v_a w_a) \quad E(pk, v_b w_b) \quad E(pk, v_c w_c)\]

Tabulation center
AddCT(pk, ...) \(\rightarrow\) \(c t_{sum}\)

Proposal passes if \(S = v_a \cdot w_a + v_b \cdot w_b + v_c \cdot w_c \geq T\)
Method 2: Homomorphic encryption

Trustees are trusted to only decrypt $ct_{sum}$

Tabulation center only accepts votes from authorized voters

Problems:

• Only supports “addition-based” votes

• What if Alice sends $ct_a = E(pk, q)$ where $q > w_a$
  • Solution: every voter attaches a ZK proof that their $ct$ encrypts an integer in $[0, w]$
  • Tabulation center drops all $ct$ with an invalid proof
Method 3: Blind signatures

What is a blind signature?

I want Sam’s signature on a secret message $m$

Public verification key: $pk$

signing key $sk$

$voter$ Alice

$m' \leftarrow \text{blind}(m)$

$\text{sig'}$

$Sam$

$Sam's\ sig\ on\ m$

$m\ ???$
Method 3: Blind signatures

choose a random nullifier $N_a$

I am Alice and I want to vote

$m' \leftarrow \text{blind}(\text{vote}_a, N_a)$

ok, you are authorized: $\text{sig}'_{\text{vrs}}$

[VOTER REGISTRATION SERVICE]

[$sk_{\text{vrs}}$]

Voter Registration Service

learns nothing about $\text{vote}_a$ or $N_a$

[tabulation center]

Accept vote if valid sig by VRS

Who just voted ??

[alice]

sig$_{\text{vrs}}$

[tor]

[\text{[vote}_a, N_a, \text{sig}_{\text{vrs}} \text{]}}$
Method 3: Blind signatures

Tabulation center collects votes in the clear

• Does not know who voted
  ... but only authorized voters can vote
  ... and only vote once \( (\text{thanks to } N_a) \)

Problems:

• Privacy of vote relies on the security of Tor

• Part-way results are public \( \text{(can be mitigated by using encryption)} \)
Back to DAOs ...
Complications

• If delegate is voting privately, how to hold them accountable?

• Clever forms of coercion are still possible:
  • Require user to turn on camera when voting on a proposal
  • Require user to place their secret key in a TEE (e.g. SGX)
The Time NounsDAO Got Private Voting

Aztec x Aragon bring private on-chain voting to Nouns

Aztec Labs · Follow
Published in Aztec · 4 min read · Mar 8

https://medium.com/aztec-protocol/the-time-nounsdao-got-private-voting-4336fe4a2c29
To set up a vote on a proposal:

• generate a random Proposal ID (PID)
• generate random encryption keys \((pk, sk)\)
• Publish \(pk\); keep \(sk\) secret  (where? will see in a minute ...)

On chain: Voting Contract (VC) has PID and \(pk\)
An encryption-based voting system

A Noun owner submits to on-chain Voting Contract (VC):

1. an encrypted vote: \( ct \leftarrow E(pk, \text{vote}) \)
2. Nullifier: \( N_{\text{Noun}} \leftarrow \text{Hash}(\text{PID, Noun-sk}_{\text{Noun}}) \)
3. a ZK proof that
   (i) they own the Noun and  (ii) \( N_{\text{Noun}} \) is computed correctly

VC accepts submission if
(i) ZK proof is valid,  (ii) \( N_{\text{Noun}} \) has not yet appeared

no double voting
An encryption-based voting system

Once voting period elapses:

- Secret decryption key $sk$ is revealed
- All votes (in the VC event log) are decrypted and tallied

How to ensure $sk$ revealed at end of vote, and not before?

- The proposal: a trusted time-lapse cryptography service

trustees that only reveal $sk$ at a pre-specified time
Complications

Voters must send vote to on-chain VC using Tor

⇒ otherwise, IP address reveals voter’s ID

If Whale voter owns 145 Nouns and submits all 145 votes at once, the voter is identified

Proposed solution: delay-relays

an off-chain service that will submit 145 votes slowly over time
Who pays the gas for casting a vote?

Voter pays ⇒ identifies voter

Proposed solution:

• **delay-relay** will pay for gas, and is then reimbursed by the Voting Contract

⇒ Voter must send votes to delay-relay over Tor
How to handle private delegation?

Unclear ...
END OF LECTURE