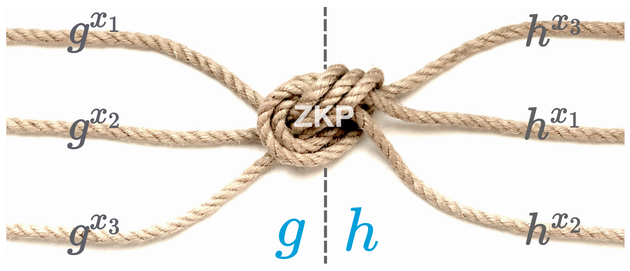
PeaceFounder: EVoting by Pseudonym Braiding

Dr. Janis Erdmanis

janiserdmanis@protonmail.ch

Braid Proof

Braiding is a cryptographic scheme that shuffles input pseudonyms $y_i = g^{x_i}$ and exponentiates them with a secret factor s, resulting in output pseudonyms $y_i' = (g^{x_i})^s = (g^s)^{x_i} = h^{x_i}$ on a new relative generator $h = g^s$. Private key owners x_i can use the relative generator h to issue cryptographic signatures with the new pseudonyms without being linked to the input pseudonyms. To ensure integrity, braiding is supported with zero-knowledge proof, represented as a knot in the image below.



The braid proof is constructed from zero-knowledge proof of shuffle and proof of decryption. Let's consider a set of members' pseudonyms $\{y_i\}$ as elements from a cryptographic group $y_i \in G$ on a relative generator $g \in G$. The braider computes a set of output pseudonyms with the following steps:

1. Generates a secret exponentiation factor s
2. Computes a new relative generator $h \leftarrow g^s$
3. Calculates ElGamal reencryption shuffle on the pseudonym set as $\{(a_i, b_i)\} \leftarrow \mathsf{Shuffle}_g(\{(h, y_i)\})$
4. Decrypts $c_i \leftarrow b_i^s$
5. Computes the resulting pseudonyms as $y_i' = c_i / a_i$

Step 3 is supplemented with a zero-knowledge proof of shuffle, whereas steps 2 and 4 with a proof of decryption. These proofs form a braid proof represented as a knot in the picture.

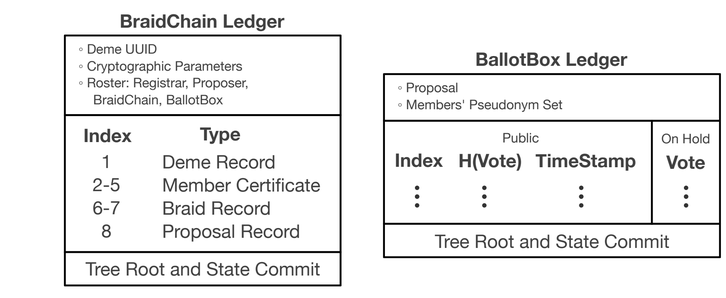
This procedure was first introduced in the work of Haeini & Spycher. Here, a slight variation of it is shown, which streamlines implementation as it avoids the identity element. A Verificatum-compatible proof of shuffle for step 3 and a custom implementation for proof of decryption in steps 2 and 4 are used, both implemented in Julia and available in the *ShuffleProofs.jl* package.

History Tree

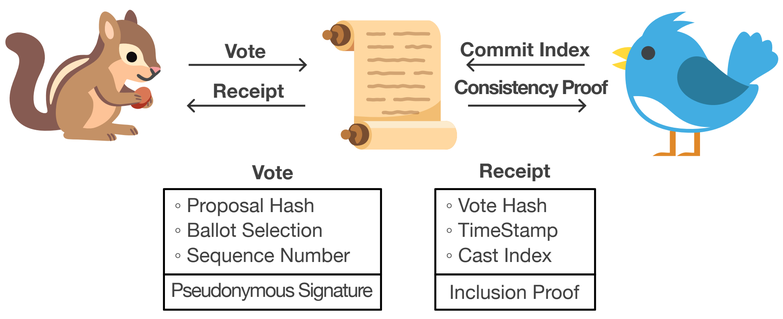
A history tree was proposed by Crosby & Wallach that enables ledger auditing without the need for complete replication. By committing to the root of this tree, the bulletin board can be held accountable for the integrity and immutability of the records.

- **Inclusion Proof:** History trees provide efficient backtracking hash chain proofs of any record's inclusion with respect to the current tree root commit. Clients get inclusion proofs along with records to verify their authenticity.
- **Consistency Proof:** This proof safeguards ledger immutability over time. It proves that a current bulletin board commit retains all records from its previous commit, with new records appended. This proof is efficient; thus, multiple clients with unpredictable queries can ensure ledger's immutability.

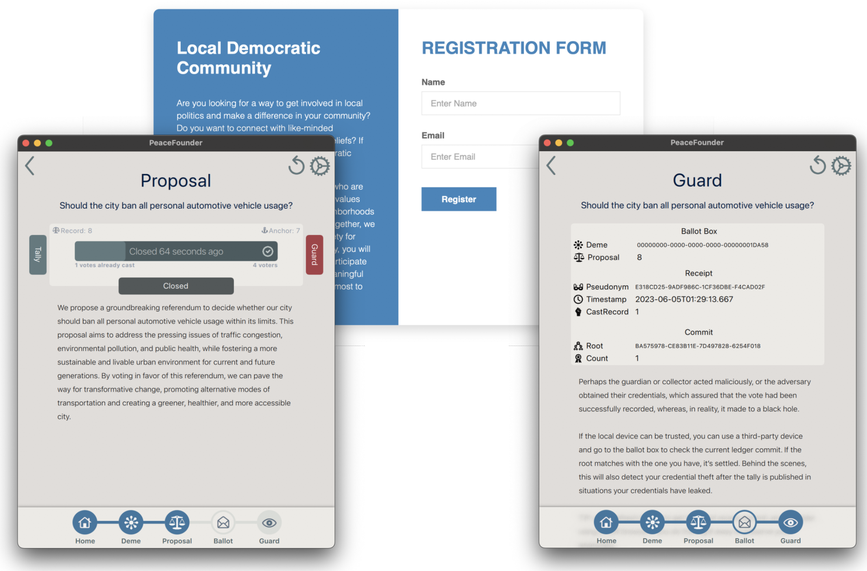
Buletin Board



- The bulletin board is split into BraidChain and BallotBox ledgers.
- For a BraidChain record to be included, it needs to be well formed and consistent with the current ledger state.
- A proposal record contains an anchor to the BraidChain ledger's state, which sets a relative generator.
- A BallotBox ledger is initialised with a proposal and corresponding members' pseudonym set, which is set by the anchor index in the proposal.



- Every vote signed by a valid pseudonym and associated with a valid proposal hash gets recorded in the BallotBox ledger, even if it is superseded or malformed. Upon recording, a receipt containing an inclusion proof is returned; if the same vote is already recorded, a receipt for it is returned instead.
- A voter keeps a consistency-proof chain and conducts incremental follow-up queries until votes are finalized. This ensures their vote's inclusion and as well votes made by others.
- The BallotBox ledger publicly displays vote hashes for integrity while concealing actual votes for fairness. This can be extended as a coercion/bribery resistance measure as, during this period, the system is receipt-free.
- A timestamp ensures that malware cannot show a receipt linked to someone else's vote. Meanwhile, a cast index helps locate the specific vote on the ledger.



Ballot Diversity & Inovation

- **Diverse Ballots:** As votes are in a plaintext signed by a pseudonym, it is trivial to support a diverse set of ballot types, including cardinal, budget planning and preferential ballots. It can also support internal whistleblowing.
- **Fluid Voting:** Pseudonymity enables fluid voting, letting voters revise their decisions within set periods during a representative term, bridging the gap between representative and direct democracy.
- **Ballot Sharding:** Long ballots can be divided into smaller shards and distributed among members' pseudonyms in a lottery. This elevates the impact of individual voters' choices, reduces the decision fatigue of a large ballot, and thus addresses the voting paradox.

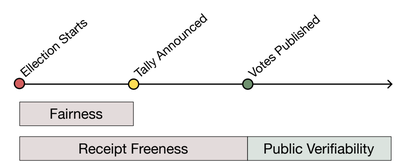
Centralised Responsability

- Built without a trusted quorum assumption, avoids threshold decryption ceremony and any trusted setup phase.
- Braiding operates on a transactional basis, making it suitable for a market. Once a braid is computed, returned and verified, there's no lingering trust assumption, allowing untrusted third parties from anywhere in the world to engage in anonymization.
- Leverages the power of voter devices to guarantee bulletin board record immutability with history tree consistency proofs. This oversight grows with member count and eliminates the need for bulletin board replication, offering both efficiency and reliability.
- Any misconduct results in publishable cryptographic evidence identifying responsible party, whether it's the registrar, proposer, bulletin board, or braider authorities.
- The system can be easily set up and managed by just one person, or can be delegated for its deployment and maintenance to a third party.

E2E Verifiable with Coercion/Bribery Resistance

	Client	Bulletin Board	Registrar	Braider
Recorded as cast	😇→❌	❌	❌	❌
Cast as intended	❌	❌	❌	❌
Tallied as recorded	❌	❌	❌	❌
Eligibility	❌	❌	😐	❌
Anonymity	😇	❌ (planed)	❌	❌ with one 😇
Receipt freeness (during moratorium)	😇	😇	❌	❌
Private key leakage detection (on device)	😇	❌	❌	❌

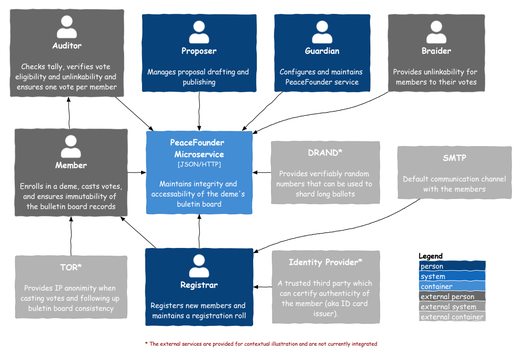
- Resistance to coercion and bribery hinges on the assumption that voters would not commit to distant positive/negative outcomes allowing them to freely revote or tag vote as coerced. We shall call this a moratorium period, within which there is a receipt-freeness but no public verifiability.



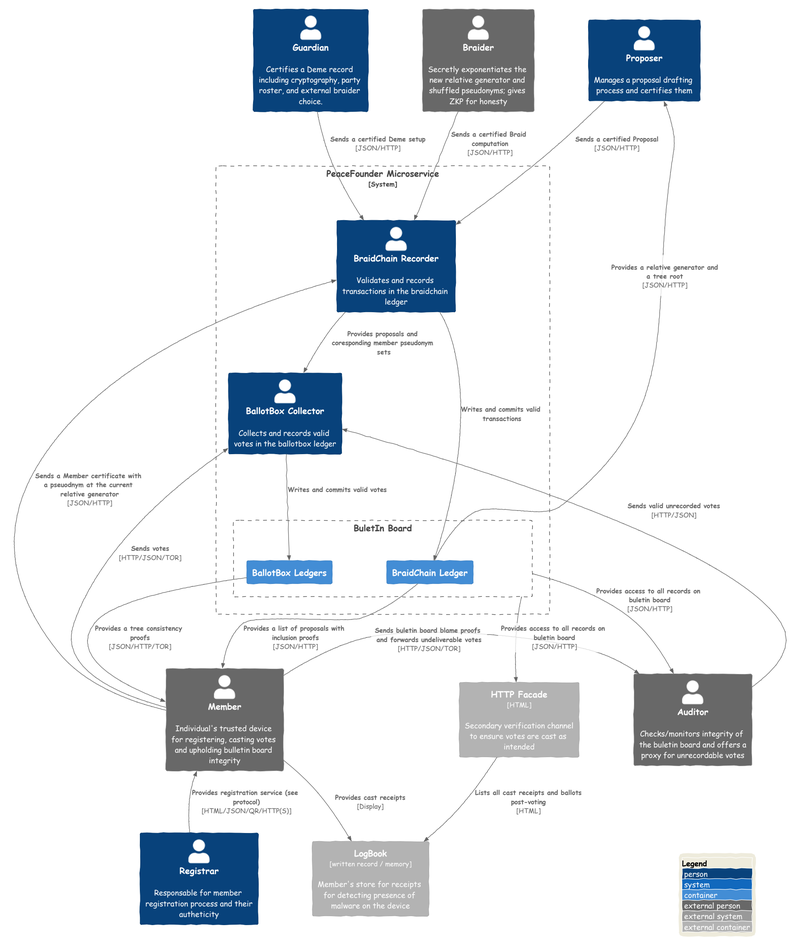
- The system is designed to resist malware and spyware. Voters can check if their vote was discarded or altered by comparing their receipt with the bulletin board's records. Additionally, the device alerts the voter if a vote is cast outside the device, which can be cross-verified with the bulletin board. Note that the aim is to enable voters to detect malware interference; it's not intended as a proof to be shown to a third party.

Responsabilities & Interactions

- Modular design and a public bulletin board allow seamless integration, facilitating custom dashboards on organisational websites.
- Custom registrars enable diverse identity authentication methods and third-party audit provisions.
- Custom proposers can adapt to varied organisational proposal submission procedures.



- Auditors uphold the integrity of both BraidChain and BallotBox ledgers, ensuring accurate vote tallying and verifying voter eligibility. Additionally, they enhance bulletin board availability by relaying votes and document suspicious activities, and can relay unrecordable votes or blame proofs between themselves to reach a consensus, paving the way for judicial actions.
- Untrusted third parties can host braiders, ensuring unlinkability for voters. Self-braiding can be useful for sanitising public evidence. Deme-based braiding exchange ensures vote privacy against dishonest organisers.
- To ensure voters' untraceability, votes are routed via the TOR service. Public IPs or domain names are recorded on the bulletin board to prevent segregation-type attacks.



Extensions

- **Proof of Participation:** A bulletin board issues blind signatures on voters' blinded identifiers included in the vote and are returned along with a receipt (assymetrically encrypted for unlinkability with a distinct key for each receipt), which voters devices unblind and can show to authority as proof of participation. After the moratorium, the signature is published on the bulletin board along with votes to avoid discrimination.

  To ensure authenticity, each pseudonym receives only one unique signature. The client device maintains a consistent, secretly and randomly generated blinding factor for all votes. As an added benefit, a bulletin board can detect private key leaks, and an invalid returned signature alerts the voter to a private key breach, promoting accountability by discouraging the use of leak-prone devices or practices.
- **Selection Asymmetric Encryption:** To maintain the imparity of auditors/proxies with respect to the votes they receive and prevent coercers/bribers from seeing what vote they receive until the moratorium ends, the ballot selections for votes can be asymmetrically encrypted. Then, a proof of correct decryption is published on the bulletin board to ensure integrity. This also enables coerced vote tagging with a decoy PIN code when a coercer/briber is observing a voter in person.
- **Ballot Sharding:** The shards of the ballot are specified, and a lottery for them is set after the proposal is published on the bulletin board. When a verifiably random salt is generated with a service like DRAND, the member pseudonyms are hashed with this salt and sorted to which the shards are allocated in order. This prevents a small minority from conspiring and allocating important questions to themselves.

References

[1] See `peacefounder.org` for roadmap, GitHub project, code and documentation, 2023.

[2] Janis Erdmanis. Zero knowledge proofs of shuffle with ShuffleProofs.jl. Presentation at the annual JuliaCon conference, 2022. Source code available on `github.com/PeaceFounder/ShuffleProofs.jl`.

[3] Rolf Haenni and Oliver Spycher. Secure internet voting on limited devices with anonymized DSA public keys. In *Proceedings of the 2011 Conference on Electronic Voting Technology/Workshop on Trustworthy Elections*, USA, 2011. USENIX Association.

[4] Björn Terelius and Douglas Wikström. Proofs of Restricted Shuffles. In *Progress in Cryptology – AFRICACRYPT 2010*, Lecture Notes in Computer Science, Berlin, Heidelberg, 2010. Springer.

[5] Rolf Haenni, Philipp Locher, Reto Koenig, and Eric Dubuis. Pseudo-Code Algorithms for Verifiable Re-encryption Mix-Nets. In *Financial Cryptography and Data Security*, volume 10323. Springer International Publishing, Cham, 2017.

[6] Scott A. Crosby and Dan S. Wallach. Efficient data structures for tamper-evident logging. In *Proceedings of the 18th Conference on USENIX Security Symposium*, SSYM'09, USA, 2009. USENIX Association.