Proof of Vote®

An end-to-end verifiable digital voting protocol using distributed ledger technology (blockchain)

Matt Becker, Lauren Chandler, Patrick Hayes, Wesley Hedrick, Kurtis Jensen, Srini Kandikattu, Pete Martin,
Scott Meier, Leopoldo Peña, Dr. Kun Peng, Aleck Silva-Pinto, Jeffrey Stern,
Liv Stromme, Lakshman Tavag, Dave Wallick, Noah Zweben

Votem Corp, Cleveland OH, USA
protocol@votem.com
www.votem.com

Abstract. We present the Proof of Vote® protocol, an end-to-end voter verifiable (E2E) digital voting system that uses blockchain to ensure the verifiability, security, and transparency of an election. The protocol leverages an ElGamal re-encryption mixnet for anonymity, a multi-signature scheme for voter authentication and authorization, and verifiable distributed key generation and verifiable decryption for vote encryption and decryption.

Keywords: End-to-end verifiability, online voting, blockchain

Proof of Vote®

TABLE OF CONTENTS

1. Introduction

1.1. Motivation

Every eligible individual should be able to actively participate in democracy by easily and safely voting when, how, and where they want. Democratic institutions generally subscribe to this philosophy, yet do little to implement it in practice. The guiding philosophy for this Protocol is illustrated below:

> "We should be guided by the dynamics of the voting public we serve – seniors whose needs include accessibility and readability of materials; persons with disabilities who have a reasonable expectation of fair and respectful service that allows for a private and secure voting experience; busy professionals who seek options for voting that match their mobile lifestyles – before and on Election Day; citizens with an array of cultural and ethnic backgrounds who depend on increased language accessibility and voter assistance; and future voters whose expectations may include things not yet considered."
>
> *Dean C. Logan*
> *LA County Registrar-Recorder*
> *August 8, 2013*

To realize this grand vision, it is impossible to envision the future of democracy where digital elections are not the global standard. Unfortunately, there exists a wide gap between this inevitability and the progress actually being made towards secure remote e-voting. In fact, most recent efforts, technically and legislatively, have been focused on in-person, paper-based voting.

As major corporate system hacks and election meddling by foreign nation states dominate the news cycle, it is no surprise that digitally enabled systems pose serious challenges. In the context of elections, these challenges revolve around reconciling the tension between strong privacy and security requirements; it is difficult to both guarantee the correctness of the results and the inability of voters to demonstrate how they have voted. Furthermore, this must all be done in an environment where everyone is potentially adversarial and no one is to be trusted.

However, moving backwards towards in-person, paper-based elections ignores an increasingly demanding, mobile, and connected society with a need for broader access, public transparency, and individual verifiability. As opposed to working collaboratively to find solutions to these problems, many are trying pull the industry back into the past. Society no longer rides horse and buggy despite the risk of auto accidents introduced by cars; the mere existence of risk should not preclude technological progress.

Consequently, in an effort to modernize the democratic process, Votem introduces our Proof of Vote® Protocol for open and collaborative comment.

1.2. Objectives

The objective of this Protocol is to help election management bodies introduce online voting with levels of security and safety that are superior to in-person, paper-based elections, with restored trust, increased access, voter verifiability, and better public transparency. This paper will explore how this is accomplished.

Specifically, the Protocol focuses on the following goals:

Restore Trust in Elections. Our current election infrastructure does not earn the trust of those using it, rather, it demands it without offering the verifiable proof that a voter's vote was cast and counted as intended. Voters must be provided sufficiently convincing evidence that election integrity has not been compromised.

Make it Easy to Vote and Impossible to Cheat. The unfortunate downside of in-person, paper-based elections is that access to voting is reduced, negatively impacting an increasingly mobile society. Mobile technology is passed its point of ubiquity; imagine if your only option for filing your taxes was to appear at the IRS in person, your only option for banking to show up to your local branch at their hours, and your only option for shopping to go to the store. It is not just a matter of convenience; people often risk their lives to participate in elections, and far too many lose their lives doing so.

Make Elections Transparent. It is inaccurate to say that paper ballots increase transparency and trust;[1] there is a distinct lack of clarity in our current voting process. When a voter casts a paper ballot in a polling place, how does she fundamentally know that her vote was properly submitted and tallied? In most elections today, even with procedural and legislative support, paper trails are not worth the paper they are printed on.[2]

In consequential governmental and organizational elections, a more transparent and publicly-verifiable process is needed. Ultimately, a method that is independently and easily verifiable by election management bodies, trusted independent authorities, and individually by each voter, is the only true solution to push democratic decision making towards greater dependability, accuracy, and accountability while restoring the trust of those participating.

1.3. Protocol Overview

We present Proof of Vote®, a protocol for conducting end-to-end verifiable elections using blockchain. Proof of Vote aims to provide irrefutable evidence of the result of a valid vote that was cast as intended and substantiated by user and third party validation for the benefit of both the voter and the elections administrative body and all other interested parties.

In some ways, our protocol is similar to other end-to-end voter verifiable (E2E) voting systems in that:

1. Voters encrypt their vote with an election-specific public key, post it to a public repository of votes, and achieve anonymity via a homomorphic cryptosystem.
2. To achieve anonymity, the set of encrypted ballots is processed via a homomorphic cryptosystem and tallied

[1] Selker, Ted, and Jon Goler. "Security vulnerabilities and problems with VVPT." (2004).

[2] Bernhard, Matthew, et al. "Public Evidence from Secret Ballots." International Joint Conference on Electronic Voting. Springer, Cham, 2017.

with proofs of correct operations.

However, Proof of Vote differentiates itself from other voting and governance protocols by being designed from the ground up to explicitly optimize for the maximum level of verifiability, accessibility, security, and transparency of an election system deployed in the real world. It offers substantial advantages over more traditional E2E systems via the use of blockchain and a multi-party signature scheme for voter authentication and authorization, aiming to be a mature and tried technological blueprint for how societies, governments, and organizations can build election systems and processes.

Blockchain architecture specifically addresses one of the most difficult factors challenging electoral integrity - the trust model. Blockchain ensures trust is distributed amongst a set of mutually distrustful parties, all of whom are potentially adversarial, that participate in jointly managing and maintaining the cryptographically secure digital trail of the election. By distributing trust in this way, blockchains create a trustless environment whereby the amount of trust required from those participating in an election is minimized.

Furthermore, Proof of Vote leverages blockchain to perform verifiable distributed key generation (for generating an election's public key), verifiable vote anonymization via mix-networks, and verifiable vote decryption. Every action that takes place as part of the Proof of Vote Protocol is realized as a transaction on the blockchain. This means that every action that takes places is verified in real-time by the entirety of the blockchain network and is inviolable once the transaction that represents that action is written to the blockchain. Furthermore, every action by a voter is fully visible to the voter and to every node at any time, maximizing visibility into an ongoing election without sacrificing the voter's anonymity.

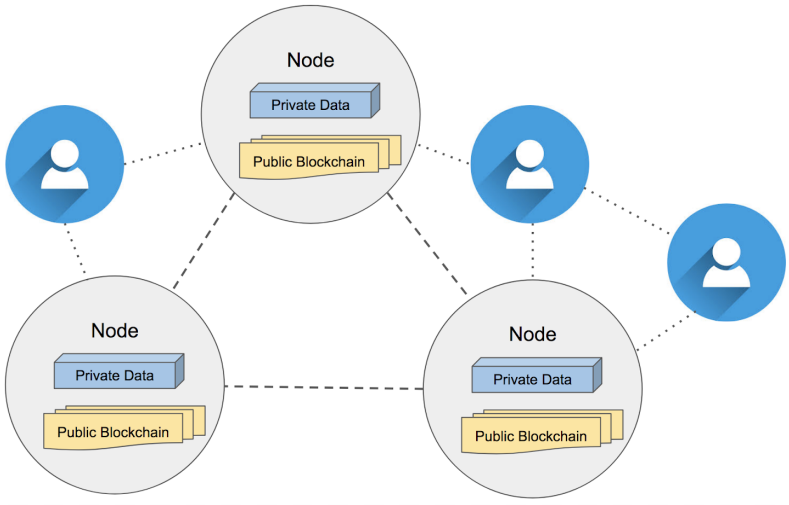

Figure 0. *Nodes communicate with each other to maintain a ledger of transactions (the blockchain). Voters interact with either all nodes or a subset of nodes. Nodes contain secret information (eg, their private keys), and also participate in a public blockchain.*

In blockchain applications every node in the network verifies every transaction written to the blockchain. This is advantageous in voting applications as it means end-to-end verification of the election happens on a rolling, ongoing

basis during the election process itself. While the use of a blockchain allows for ongoing verification of an in-process election, it also allows for a retrospective end-to-end verification of all election activities after the election has finished. Elections using Proof of Vote could be recreated in context and verified indefinitely after the end of the election by validating the results and all signed vote transactions.

This real-time auditing stands in contrast to an evolving "public verification" method known as "Risk Limiting Audits" (RLA). RLA's are well intentioned, but they do not instill confidence in the integrity of an election to the degree that is required. Because these audits are only conducted post-election, they only verify that a ballot was counted, not that the collection of ballots was handled properly, was cast as intended, nor individually correctly tallied. Retrospective audits alone fail to secure elections as they occur; Enron and other fraudulent "audited" organizations are testaments to this risk. The aphorism holds, "garbage in, garbage out," which is why real-time validation checks, in contrast to retrospective checks, are so critical.

In the Proof of Vote, an election is run on a blockchain network, where each node participating in the network is explicitly granted a set of permissions by the authority conducting the election. These permissions include the ability to produce blocks, participate in the blockchain consensus algorithm, notarize blocks, be an election-key trustee, and/or be a mix-network node.

Each node in the network may have one or more permissions and each participating node holds a public/private keypair, which is used to identify the node and participate in the blockchain. Potential node operators may request permissions and participation rights from the Election Authority, the legal entity responsible for the election. These parties can participate as node operators across multiple elections or be used as additional nodes for individual elections on a case-by-case basis determined by entity interest and the election authority's determination of need. Voters, on the other hand, do not operate nodes, but instead interact with the blockchain by submitting data for block creation to the chain at various steps in the voting process (for example, identity authentication, voting, and vote verification).

2. Proof of Vote™ - An Overview

In this section we describe the protocol chronologically as a series of steps.

1. Election creation
2. Key Generation
3. Pseudonym creation
4. Authentication
5. Authorization
6. Initialization
7. Ballot Delivery
8. Ballot Marking
9. Vote Encryption
10. Vote Submission and Storage
11. End of Voting
12. Vote Anonymization
13. Vote Decryption
14. Election Finalization
15. Tally

A fuller analysis and discussion on components of the steps laid out above is available in Section 3 - Discussion and Analysis.

Step 1. Election creation

Election creation involves an Election Authority defining the content and rules for an election. Once finalized, the election is locked making static Election Definitions, which are hashed and signed by the Election Authority, providing a pre-commitment by the Election Authority on the content of the election.

Election creation involves the following sub-steps:

1. The Election Authority produces a set of election content and rules that define the election. As part of this step, the Election Authority defines a list of Ballot Styles for the election. A *Ballot Style* is an ordered list of contests and those contests' candidates and measures, plus any additional marking and/or tabulation rules needed for a given contest (for example, whether or not a contest supports write-in candidates). A Ballot Style is assigned a unique ID and is made up of two component parts, each assigned unique IDs:

    a. A *Marking Style*, which contains various language localizations for the Ballot Style, any supporting audio and image content, candidate ordering, and a unique ID.

    b. A *Ballot Submission Style*, which defines the rules for what constitutes a valid vote for this Ballot Style. It would generally contain a list of candidates, if a write-in is allowed plus the number of selections allowed

on a contest by contest basis, and a unique ID.

2. Election Officials initialize the ***Election Creation Transaction*** **(TRANSACTION 0)** on the blockchain. This transaction contains:

   a. Election Rules including dates, times, and tally rules.
   b. List of all Ballot Style IDs and associated marking rules.
   c. Public keys of designated trustees, mixers, authenticators, and authorizers.
   d. Parameters for election public key cryptography.
   e. Data-retention policy for private keys used by trustees in this election.
   f. Signature(s) on the above by the Election Authority and Election Officials.

Step 2. Key Generation

Proof of Vote employs an ElGamal[3] encryption scheme with distributed key generation and threshold encryption and decryption functions. It uses verifiable key sharing to generate and share private keys amongst trustee nodes that jointly hold the private key for the election. This distributed generation and holding of private keys is important because no single trustee node will individually have the power to decrypt encrypted votes; at no point in time will a whole private key ever exist under the control of a centralized node, meaning the power of decryption has been decentralized (unless otherwise determined by the Election Authority);

Each trustee node produces a ***Key Generation Transaction*** **(TRANSACTION 1)** which contains:

1. A private key portion, encrypted with the public key of a subset of all trustees.

2. The public portion of the generated key.

3. A public commitment that can be used to verify the correctness of the key share.

After a trustee receives all *Key Generation Transactions* that include its public key, the trustee verifies that all the keys given sum to a private keyshare that recovers an unknown but unique private key. Once this is done any trustee may generate a single ***Election Public Key Transaction*** **(TRANSACTION 2)** which contains the public key for the election and jointly confirms that the unknown private key for the election is correct.[4] Only one *Election Public Key Transaction* is produced per election and once the transaction is in place, the blockchain is in a state to receive voter information and votes.

See Section Key Generation for more information.

Step 3. Pseudonym Creation

Voters are identified by a pseudonym which is unique to them, and only used in a single election. These pseudonyms

[3] T. ElGamal (1985). "A public key cryptosystem and a signature scheme based on discrete logarithms" (PDF). IEEE Trans Inf Theory. 31
[4] Cortier, Véronique, et al. "Distributed elgamal á la pedersen: application to helios." Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society. ACM, 2013.

are used to provide partial, but not full, anonymization[5]; because they are tied to a voter's public key, they facilitate the handling of voters in the Protocol.

1. Each voter, using the voting application on their connected device, creates a public/private RSA key-pair.[6]
2. The public key is hashed using SHA256.[7]
3. The hash is hexadecimal-encoded to a human readable string. This string is the voter's pseudonym, which uniquely identifies the voter for the duration of the election but is meaningless to any viewer without the knowledge of how it ties to an individual voter.

See Pseudonyms for more information.

Step 4. Authentication

Authentication is the process by which a voter proves their identity - that a person is who they say they are. The Protocol does not specify what the voter uses to prove their identity, since this will vary from election to election, but rather prescribes a generic multi-signature scheme whereby a set of authenticators jointly authenticate the voter in a distributed and redundant manner.

Strong and secure authentication is a requisite for discerning voter eligibility in most statutory elections. In remote environments, there exists a different set of challenges for an election authority to properly identify voters than in traditional voting contexts, particularly as any implemented system has to work within the confines of that jurisdiction's laws and regulations for determining voter eligibility. It is ultimately the Election Authority's responsibility to maintain a voter registry and determine the set of voter eligibility rules and mechanisms to authenticate voters against those rules. Proof of Vote's Authentication details how determining eligibility works provided the Election Authorities rules and mechanisms.

In the Proof of Vote Protocol, an authenticator is an authority delegated by the Election Authority to verify the identity of a voter. Example of authenticators include the Election Authority itself, DMVs (Departments of Motor Vehicles), passport offices, or any other entity that can assuredly match identifying information to a voter.

To be authenticated, the voter sends their pseudonym to a quorum of authenticators to be signed, along with identifying information and any additional information required to prove the veracity of the identifying information. Each authenticator independently authenticates the voter, and produces two signatures:

1. A signature of the voter's pseudonym.

---

[5] Full anonymization is achieved upon Step 12. Vote Anonymization in which Vote Transactions pass through re-encryption mixed networks implemented by the Protocol. Partial anonymization refers to the fact that authorized nodes can tie voters to pseudonyms during the steps of Authentication and Authorization, the authorized node tied to a Vote Transaction can tie a pseudonym to an IP address, and that timing attacks are conceivable prior to full anonymization by authorized nodes.

[6] Rivest, Ronald L., Adi Shamir, and Leonard Adleman. "A method for obtaining digital signatures and public-key cryptosystems." Communications of the ACM 21.2 (1978): 120-126.

[7] An approved NIST hash algorithm: Dang, Quynh. Recommendation for applications using approved hash algorithms. US Department of Commerce, National Institute of Standards and Technology, 2012.

2. A signature of a package that contains the voter's pseudonym and identifying information.

Each authenticator records a receipt of the transaction privately for auditability. The voter then combines these signatures such that the voter now has a pseudonym signed by a quorum of authenticators and an identifying-information/pseudonym package signed by a quorum of authenticators. This quorum can contain as few as one authenticator or as many as the Election Authority deems necessary. Additionally, rules can be set around authenticating agencies such that a certain type is always required and some subset of others is needed to reach quorum, or other such combinations.

See Authentication for more information.

Step 5. Authorization

Authorization is the process by which an authenticated voter is authorized to vote and has their Ballot Style ID assigned. Procedurally, it operates very similarly to external authentication.

Authorizers are entities that can match a voter's identifying information to a Ballot Style. In many cases there is only a single authorizer (the Election Authority itself, in possession of the voters list). However, Proof of Vote implements a multi-signature scheme for authorizers to allow for distributed and redundant authorization in cases where Election Authorities want to distribute trust and responsibility for the voters list management.

To authorize, the voter sends their authenticator-signed identifying-information/pseudonym package to a quorum of authorizers. The authorizers check the signatures, check to make sure that the voter identified by the identifying-information is allowed to vote, and check what Ballot Style ID they are assigned. Each authorizer produces two signatures which are sent back to the voter:

1. A signature of the voter's pseudonym.
2. A signature of a package that contains the voter's pseudonym and Ballot Style ID.

Each authorizer stores a log of the transaction for auditing purposes. The voter then combines these signatures such that the voter now has a pseudonym signed by a quorum of authorizers and a Ballot Style ID/pseudonym package signed by a quorum of authorizers.

See Authorization for more information.

Step 6. Initialization

Up to this point, a voter has had no interaction with the blockchain; this Initialization step is important because it allows subsequent voter transactions to be tied to their original Initialization attestation on the blockchain. The voter

is now in possession of all the information required to produce an ***Initialization Transaction*** **(TRANSACTION 3)** on the blockchain. The voter packages the following information in their voting application to produce an *Initialization Transaction*:

1. Voter's pseudonym
2. Voter's RSA public key
3. Authentication pseudonym signatures
4. Authorization pseudonym signatures
5. Ballot Style ID/pseudonym package signed by a quorum of authorizers..
6. A signature on all of the above information using the voter's RSA key.

The voter submits this transaction to the blockchain and waits for confirmation. Blockchain nodes will validate the above information establishing that a voter's pseudonym corresponds with a valid authorized voter and that a voter's identity has been verified by a quorum of accepted authentication services, and ultimately include the transaction in a block.

Step 7. Ballot Delivery

Once the voter has been initialized, they are ready to request a ballot. The objective of this step is to deliver the correct ballot to the authenticated and authorized voter. The voter device application randomly selects a node on the blockchain permissioned to provision ballots and requests an unmarked ballot. The node sends the user an unmarked ballot. The delivered unmarked ballot contains:

1. Ballot Style
    a. Ballot Style ID
    b. Marking style
    c. Marking style ID
    d. Submission Style
    e. Submission Style ID
2. Election Public Key created by trustees

The Voter checks the *Election Creation Transaction* to make sure the fingerprint on the Ballot Style and the associated Submission Style ID are all correct which validates to the voter that they have been served the correct Ballot Style for the current election. This step is defined separately from other steps in the process because it emulates traditional election process and facilitates the spoiling of ballots through ***Revocation Transactions*** **(TRANSACTION 4)**.

The *Revocation Transaction* allows the nullification of an *Initialization Transaction*. It requires that the voter's identity be verified by a quorum of accepted authentication services. This transaction can be necessary when a voter chooses to spoil their ballot or when a voter misplaces or loses possession of their private key. The Revocation

transaction will fail validation if a vote has already been cast by the affiliated pseudonym. The transaction includes:

1. Pseudonym of the Voter.
2. Pseudonym/Revocation combination signed by a quorum of accepted authentication services.

See Ballot Delivery for more information.

Step 8. Ballot Marking

Once in possession of an unmarked ballot, the voter is ready to mark their ballot on their device. The voting device will read the unmarked ballot and display an interface to allow the voter to mark their ballot on a contest by contest basis. The voting application will follow best practices with regard to usability and accessibility as well as multi-language implementation as prescribed by the Election Authority running the election.

Step 9. Vote Encryption

The voter, via their voting device, encrypts their vote using ElGamal with the trustee's public key associated with the election. The encrypted vote is made up of two concatenated components:

1. The ballot Submission Style ID
2. The ballot-markings denoting the vote itself

The voter's device then generates a Zero Knowledge Range Proof [8] that the first component of the encrypted ballot is the ballot submission style ID assigned to the voter in the Ballot Delivery step. Once the voter's device has encrypted the vote, the voter is able to challenge the trustworthiness of their device via interactive proofs of device challenges.[9]

See Vote Encryption and Interactive Device Challenges for more information.

Step 10. Vote Submission and Storage

The voter's device creates a ***Vote Transaction*** **(TRANSACTION 5)**, composed of the following:

1. Voter's pseudonym
2. The encrypted ballot
3. The Zero Knowledge Range Proof of correct ballot-submission style ID
4. The ballot-submission style ID

[8] Kun Peng and Feng Bao. Batch Range Proof for Practical Small Ranges. AFRICACRYPT 2010, LNCS6055, pages 114-130.
[9] Josh Benaloh. Simple verifiable elections. In EVT, 2006.

5. The voter's signature on all of the above

The voter's device then submits this *Vote Transaction* to a node to be written to the blockchain, and waits for confirmation of the *Vote Transaction*, which is meant to act as the secure and immutable process of casting a vote. The transaction contains a zero knowledge proof which can be interpreted as a coercion resistant methodology providing the voter verifiability of their cast vote without revealing the vote itself. The ballot style ID is also contained within this transaction to ensure the correct ballot was marked for the corresponding pseudonym.

See Vote Submission and Storage for more information.

Step 11. End of Voting

After voting concludes at the end of the alloted election time, as denoted in the *Election Creation Transaction*, any authorized node produces an ***End Of Voting Transaction*** **(TRANSACTION 6)**. After this transaction is written to the blockchain, no more blocks containing *Vote Transactions* will be accepted.

Step 12. Vote Anonymization

Vote anonymization is an optional step that occurs after all votes are cast and before votes are decrypted for tallying; Proof of Vote employs a re-encryption mix network to fully anonymize votes. Mixing happens in stages, with each mixing node shuffling and re-encrypting a batch of votes in order. As the batch go through each stage, the shuffling mixer node produces a ***Shuffle Transaction*** **(TRANSACTION 7)** composed of:

1. The Batch ID, which is the SHA256 hash of all input vote ciphertexts for the batch
2. The stage of mixing (ex: first shuffle for a batch is stage 1, second shuffle is stage 2, etc.)
3. A batch of re-encrypted and shuffled votes.
4. A zero knowledge proof of correct re-encryption and shuffle.

Votes are assigned to equally sized batches. Batches are assigned to a mix-network composed of mixers that are permissioned to participate in the mix-network. At the end of the re-encryption shuffle process, all votes are anonymized, assuming there is at least one honest mixer for each mix-network.[10] [11]

See Mix-Networks for more information on implementation.

[10] Kun Peng. Anonymous Communication Networks: Protecting Privacy on the Web. 2014 by CRC Press/Taylor & Francis Group.
[11] Kun Peng. Assumptions And Conditions For Mix Networks --- Analysis, Clarification And Improvement. International Journal of Security and Networks 9(3): 133-143 (2014).

Step 13. Vote Decryption

After votes have been anonymized, votes are ready to be decrypted. Every trustee processes all encrypted votes, creating a ***Partial Decryption Transaction*** **(TRANSACTION 8)** for each vote, composed of:

1. The partially decrypted vote; a partial decryption of the vote ciphertext $s_j = a^{d_j} mod\ (p)$ where d_j is the trustee's private key share.
2. A zero knowledge proof of correct partial decryption.

After a quorum of enough trustees have partially decrypted a given vote, any node may produce a ***Vote Decryption Transaction*** **(TRANSACTION 9)**, fully decrypting the vote and writing it to the blockchain. *Vote Decryption Transactions* occur if a set S of at least t trustees provide a *Partial Decryption Transaction* for a vote, such that the vote plaintext $s = \frac{b}{\prod_{P_t \in S} s_t^{u_t}}$ can be recovered where $u_i = \prod_{P_j \in S, j \neq i} \frac{j}{j-1}$.

Step 14. Election Finalization

After all votes are decrypted, any node may produce an ***Election Finalization Transaction*** **(TRANSACTION 10)**. This transaction formally ends the vote processing portion of the election and is composed of a SHA256 hash of all transactions in an election (which can be used to do a full end-to-end verification), and a separate SHA256 hash of all vote plaintexts in an election (which can be used to securely tally without doing a full end-to-end verification). *Election Finalization Transactions* denote a guarantee that all nodes agree that the election is complete and that the given set of transactions fully comprises the end-to-end-verifiable set of transactions that make up the election.

The transaction therefore contains:

1. A SHA256 Hash of all transactions for the election.
2. A SHA256 Hash of all vote plaintext as contained in *Vote Decryption* transactions.

This final transaction should be widely distributed via side-channels to allow independent verification of the hash that fully defines the election. Upon confirmation of the *Election Finalization Transaction*, all trustees refer to the data-retention-policy in the *Election Creation Transaction* and optionally destroy their private keyshare, providing limited forward election integrity.

Step 15. Open Tally

In the Proof of Vote, tallying happens after the election is finalized and, because all votes are decrypted, may happen outside the context of the blockchain. Anyone may independently tally the results - including election officials to produce official results. Because votes are fully decrypted in order to tally, arbitrary tally methods are supported; this provides a significant advantage over homomorphic tally schemes where only a subset of tally methods that are

amenable to homomorphic operations are supported.

When an election is finished, a final *Election Finalization Transaction* is written to the blockchain. To tally, a tallier will first obtain a copy of vote plaintexts from all *Vote Decryption Transactions* and ensure that the hash of these votes is correct. The tallier is then free to tally all results from the decrypted votes and publish the results.

End-To-End Verification

While the Proof of Vote performs a real-time “rolling validation" of an election as transactions are written to blocks, a full end-to-end verification of a complete election is also possible. End-to-end verification of the protocol involves downloading all transactions for an election, stepping through each transaction, and verifying all transactions in order. The final transaction that makes up an election contains a hash of all transactions, which can be used to verify that all transactions are accounted for.

To perform an end-to-end verification of a complete election, all transactions that make up an election are verified in order. A high-level synopsis of the verification steps is as follows:

1. Download a copy of all transactions for an election. This download might be obtained via nodes participating in a network, or if the election is long past, from an Election Authority who has kept a copy in their digital archives.
2. Compute the hash of all transactions and compare it against the hash published in the *Election Finalization Transaction*. If the same hash is available via side-channels, it should be compared against these as well.
3. Verify that the Election Public Key has been properly generated by verifying all *Key Generation Transactions* and verifying the single *Election Public Key Transaction*.
4. If a log is available from Authenticators, verify that there are not more *Initialization Transactions* than unique Authentication events.
5. If a log is available from Authorizers, verify that there are not more *Initialization Transactions* than Authorization events.
6. Verify that all *Initialization Transactions* contain a unique pseudonym, and that a quorum of Authenticators and Authorizers have signed the pseudonyms. Verify that a quorum of Authorizers have signed the combined pseudonyms/Ballot Style ID.
7. Verify that all *Vote Transactions* are associated with a valid (non-revoked) pseudonym and are properly signed. Verify that the *Vote Transaction* contains the correct Ballot Style ID and that the Zero Knowledge Proof proves that the Submission Style ID is encoded in the encrypted vote.
8. Verify that no *Vote Transactions* are on the blockchain after the *End Of Voting Transaction* block.
9. Verify that all votes have been properly shuffled in the mix network. Verify that votes have been properly batched to the correct mixers and that each mixer has produced valid *Shuffle Transactions*.
10. Verify that all votes have been properly partially decrypted. Verify that a sufficient number of trustees has produced a *Partial Decryption Transaction* for every vote and that these transactions prove that the vote was correctly partially decrypted.
11. Verify that each vote has been correctly decrypted via a *Vote Decryption Transaction.*

12. Verify that the *Election Finalization Transaction* correctly computes the SHA256 hash of both all transactions and all *Decryption Transactions*.

13. Tally all votes and verify that the official published results match the tallied results.

In practice, Proof of Vote can greatly improve voter confidence in an electoral process by introducing transparency and verifiability into every step of the process. Proof of Vote fulfills the requirements of an end-to-end verifiable online voting system by providing irrefutable evidence of the result of a valid vote that was cast as intended and substantiated by voter and third party validation while protecting ballot secrecy. It accomplishes this with the following properties:

1. Every voter can verify that their own vote has been counted correctly by verifying that it is on the blockchain.

2. Every voter can verify that everyone else's ballots have been counted correctly by verifying that vote-anonymization and vote decryptions transactions are valid.

3. Skeptical voters do not have to trust that their device has encrypted their ballot correctly. Instead, every voter can challenge and check the integrity of the process ensuring that their device has encrypted their ballot correctly through the use of a vote-verifying challenge. With a statistically sufficient percentage of auditors performing this validation, the integrity of votes cast can be probabilistically guaranteed.

4. In the normal case, a voter cannot prove how they voted to any third party due to vote anonymization.

5. Every voter can verify that all ballots have been cast by a voter that has been authenticated and authorized to do so. Auditors with access to authorization and authentication logs can verify that authorization and authentication has been done correctly.

6. Use of a blockchain ensures that the election is durable and is always in a fully validated state.

3. Discussion and Analysis

3.1. Blockchain

Blockchains are digital ledgers in which transactions are recorded chronologically and are verified by a network of mutually distrustful nodes. These independently verifiable transactions are logically grouped into ‘blocks’. Each newly created block holds a reference to the cryptographic hash of the previous block, bestowing the property of immutability to the entire chain.

3.1.1. Rationale

Blockchains have several advantages over centralized datastores; the most pertinent to elections are:

1. **Decentralized Trust:** As each node in the network independently validates every transaction in proposed blocks before a commit, trust in the proper operation of the protocol is spread throughout the entire network instead of being centralized into a single entity. One important property of blockchains is the ability to retain integrity in the face of multiple compromised or malicious nodes; in the context of elections, it is often the base presumption that some actors are adversarial, compromised, or malicious.
2. **Immutability:** As each block is committed to the blockchain, a reference to the previous block’s merkle hash prevents tampering after the commit.
3. **Verifiability:** By giving independent auditors the ability to add their own network, and independently validate the transactions, the blockchain network allows for a higher degree of transparency than traditional centralized datastores without sacrificing the security of the datastore.

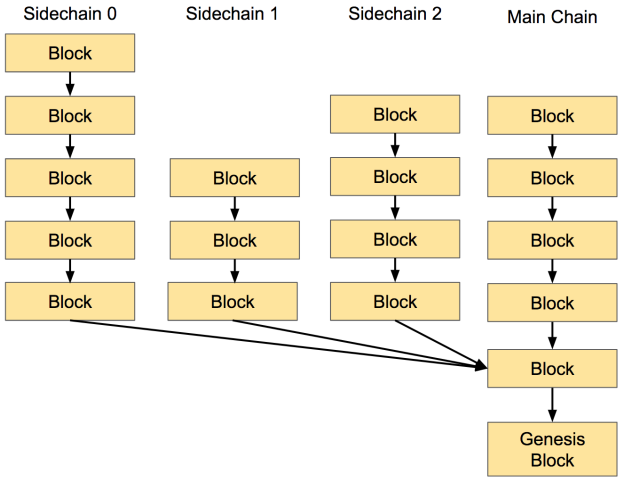

Figure 1. *Blocks of transactions are arranged in a chain, with each block referring to the block that precedes it. Every chain originates from the Genesis Block, and is partitioned into shards. Shards branch off the main chain as a set of sidechains.*

Votem uses blockchain in order to implement a verifiable and transparent voting system. By applying the pertinent properties of blockchains to voting, Votem’s protocol is able to guarantee:

1. **Independently Verified Results**: In traditional paper ballot voting, the voter must trust the poll workers and local election officials to ensure that their vote is recorded. Blockchain networks allow for multiple nodes to independently ensure that the vote is recorded.
2. **Vote Accuracy Immune from Compromise:** In traditional paper ballot voting, the voter must trust the poll workers, and local election officials to ensure that their vote is not tampered with. Blockchain networks enforce immutability through merkle trees and chained references.
3. **Public End-To-End Verifiability:** In traditional paper ballot voting, the voter has no means to verify that their vote is counted in the final tally. Blockchain allows voters, as well as independent auditors, to independently verify that their vote has been collected and will be included in the final tally.

3.1.2. Node Permissions

Nodes in the Proof of Vote blockchain network operate on a proof-of-authority[12] basis. Proof-of-authority does not depend on the nodes solving difficult mathematical problems (proof-of-work[13]), nor on the nodes proving they have a currency (proof-of-stake[14]), but rather are assigned permission to operate on the network by a central authority. In the Proof of Vote implementation, the Election Authority is the ultimate arbiter of who can operate nodes on the network. Node operators could potentially include election administrators, designated election observation/safeguard organizations, academic institutions or universities with special interests in elections, voter advocacy groups, governmental entities interested in oversight and/or election participation, and even private bodies such as political parties with significant interest in any given election.

Permission Hierarchy

Proof of Vote defines the following node permissions:

- Trustee
- Mixer
- Consensus Participant
- Block Notary
- Public Verifier

Any one node may have any combination of permissions. Permissions for trustees and mixers are defined as part of the *Election Creation Transaction*; Consensus Participants, Block Notaries and Public Verifiers are permissioned as part of the underlying blockchain protocol. Election Officials are responsible for generating and certifying the *Election Transaction*, and may be involved in granting or revoking permissions for nodes during an in-progress election, but are not nodes with permissions themselves. In this paper we present Trustees and Mixers as permissioned nodes participating in the blockchain for simplicity, but in reality they may operate independently of the blockchain itself, submitting trustee and mixer transaction, but not necessarily participating in consensus.

[12] https://en.wikipedia.org/wiki/Proof-of-authority
[13] https://en.wikipedia.org/wiki/Proof-of-work_system
[14] https://en.wikipedia.org/wiki/Proof-of-stake

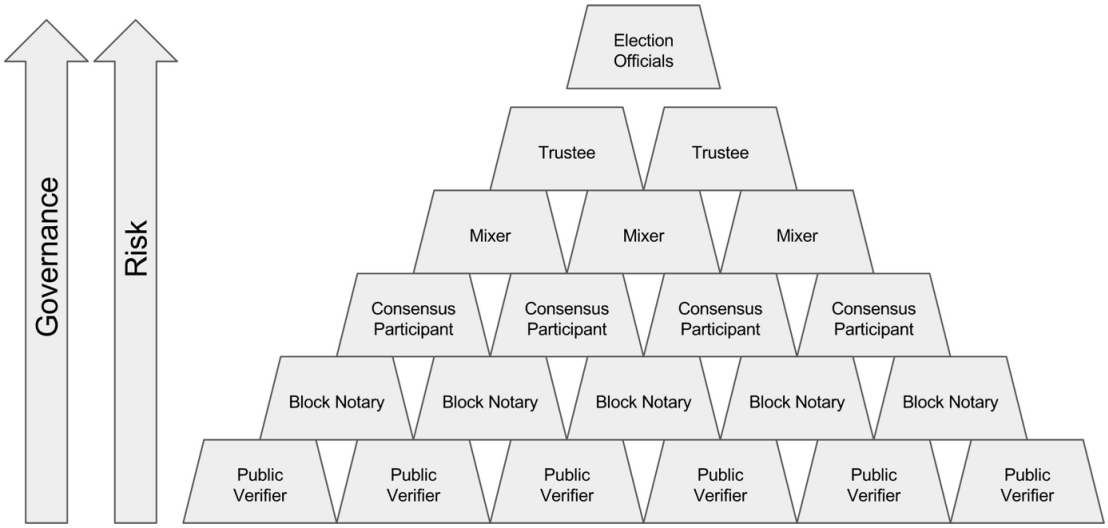

***Figure 2**. The Proof of Vote Protocol defines various sets of permissions for nodes participating in the network.*

Node Permission	**Description**
Trustee	Trustee Nodes with the "Trustee" permission are responsible for collectively holding the private key used to encrypt votes. They are also jointly responsible for decrypting votes. Before voting starts the trustees will jointly perform a key generation ceremony and publish the collective public-key as part of the *Election Public Key Transaction*. *See the Appendix for more information on the security of threshold-encryption, encryption, and decryption.*
Mixer	Nodes with the "Mixer" permissions are available to participate in the mixnet. Given the high computational requirements needed to participate in the mixnet, nodes with the Mixer permission should be provisioned with ample performance characteristics to fulfill this function. Mixer node misbehavior is controlled by requiring mix nodes to prove correct mixing via a Zero Knowledge Proof of correct shuffle.
Consensus Participant	The main responsibility of appending blocks to the blockchain falls on consensus participant nodes. Consensus participant nodes receive various transactions submitted by the voting client, mixing nodes, and trustee nodes and have the responsibility of performing validation on the incoming transactions and then broadcasting these transactions to the rest of the consensus participant nodes. The rest of the consensus participant nodes then perform validation on the submitted transaction and proceed to vote on the transaction. The sum of these consensus participant votes collectively accept or reject transactions to be appended to the blockchain.
Block Notary	The Block Notary role permits nodes to be involved in the consensus process by signing off on consensus participant's approved transactions before being appended to the blockchain. Block notaries have no voting weight on proposed blocks but can see all voting transactions

	performed by consensus participants and verify all transactions. Block notaries can also be promoted to consensus participants in the case of repeated compromise of consensus participant nodes, consensus participant node failure, or severe network load.
Public Verifier	In the Proof of Vote protocol, the ability to support different levels of access to nodes is paramount. Trust and verifiability are at the center of elections and to facilitate this in a blockchain architecture it is critical to include public verifiers. These public verifier nodes are granted read-only access transactions on the blockchain. Public verifier nodes have access to all blocks and transactions but are not privy to any of the internal voting mechanisms of consensus participants.

3.1.3. Sharding

On public blockchains, the addition of nodes logarithmically increases the latency of transactions and confirmation times for each block. Proof of Vote uses a model where only a limited number of trusted nodes will be allowed to propose and vote on blocks. While additional nodes can audit each transaction, this can be done in a way that does not affect throughput or confirmation time.

Sharding can be an effective strategy to improve the transaction latency and throughput on a blockchain.[15] Sharding is a partitioning mechanism by which compute and storage can be logically distributed across different hardware (or virtual hardware) in order to improve performance. In an architecture with *N* shards and perfectly distributed traffic to each of the shards, a single shard would only have to accomplish *1/Nth* of the total work. Traditional blockchains require that every transaction is available for audit on every node in order to address the double spend problem[16] and can pose a challenge when trying to scale the throughput of the network. Proof of Vote uses a pseudonym-based transaction system which allows for sharding as a first class citizen of its blockchain implementation. In voting applications, protection against double spend only needs to be prevented on a per voter basis; the validity of one vote is not dependent on the validity of another vote.

[15] A. Gencer, R. van Renesse,and S. E. Data, “Short paper: Service-Oriented sharding for blockchains,” Financial Cryptography and Data ., 2017.
[16] https://en.wikipedia.org/wiki/Double-spending

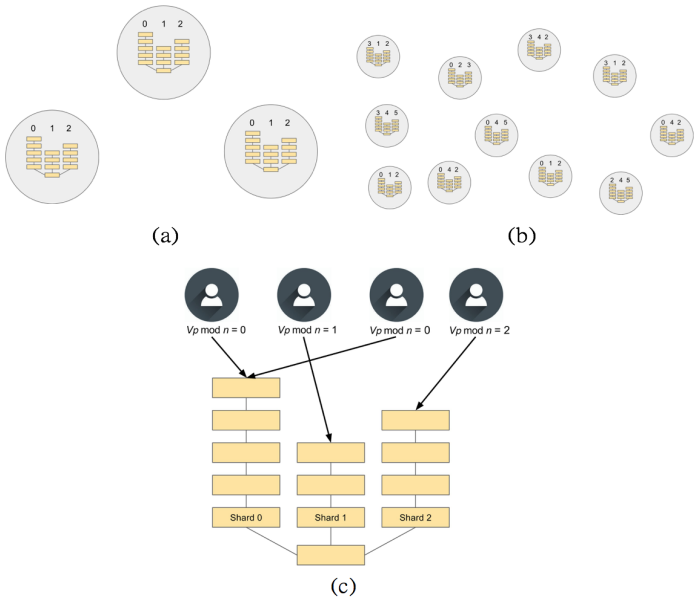

Figure 3.

(a) On a small network, all nodes hold all shards.
(b) On a large network, shards are assigned to a subset of nodes.
(c) Transactions are assigned to shards by pseudonyms according to equation (1).

In Proof of Vote, sharding is accomplished by assigning transactions to either the *main chain* or a *sidechain*. All nodes participate in the main chain, while a subset of nodes participate in sidechains. Alternatively, a node may participate in all sidechains, but process each sidechain concurrently. The *Election Creation*, *Key Generation*, *Election Public Key Generation*, and *Election Finalization* transactions all take place on the main chain. All other transactions may take place on sidechains.

In Proof of Vote, all voters generate an RSA key pair. The public key is hashed using SHA256 to generate the voter's pseudonym, which is used to uniquely identify the voter for a single election (*see Pseudonym for more details on pseudonym generation*). For this sharding strategy to work, all pseudonyms need to be uniformly distributed across the space of all possible pseudonyms. This is acheived by using SHA256, which (approximately) uniformly maps inputs to the output space,[17] resulting in pseudonyms being uniformly distributed across the space of all possible pseudonyms. To assign voters and their transactions to a shard, the following function is applied:

(1) $S_n = V_n \ mod(n)$ where S_n is the shard assignment, V_n is the voter's pseudonym, and n is the total number of shards (Figure 3(c)).

One shortcoming of this method is that by iteratively regenerating their public key (and therefore their pseudonym), technically capable voters would be able to choose which shard their transactions take place on. We do not consider this a significant risk.

[17] Michiel Buddingh. The distribution of hash function outputs. Available at https://michiel.buddingh.eu/distribution-of-hash-values

Mixing and decryption related transactions would be written to a sidechain, and would operate only on the votes written to the same sidechain.

On small networks with under one hundred nodes, we would expect every node to hold a copy of all shards (Figure 3(a)). All nodes will process all transactions, with sharding providing improved block confirmation time and transaction latency, but not reducing the amount of validation that all nodes must perform. On larger networks, each node would be responsible for a subset of shards (Figure 3(b)), with each shard being assigned a set of nodes.

3.2. Components

3.2.1. Key Generation

Proof of Vote employs an ElGamal encryption scheme with distributed key generation and threshold encryption and decryption functions. Each of w trustee private key holders shares his private key portion among the others such that any t of them can reconstruct the private key. Every trustee sums up the key shares for him such that any t of them can reconstruct the sum of all the trustees' private keys, which is the vote decryption key and its corresponding public key is the vote encryption key. t-out-of-w threshold ElGamal is generated and used in a distributed pattern as follows:

1. **Parameter Setting:** G with order q is a cyclic subgroup of Z_q^* where q and $p = 2q + 1$ are large primes, g is a generator of G. Parameters p, q, g, y are published as part of the *Election Creation Transaction*. There are w trustees $P_1, P_2, ..., P_w$ who cooperate to manage the threshold ElGamal cipher.
2. **Key Generation:** Each P_i chooses his private key seed $x_i \in Z_q$ and publishes $y_i = g^{x_i} mod(p)$ for $i = 1, 2, ..., w$ as part of the *Key Generation Transaction (Transaction 1)*. One transaction per trustee is produced, and is the first step in generating a shared Election Public Key. First all trustee's public keys are sorted and each trustee is assigned an integer for $i = 1, 2, ..., w$, where w is the number of trustees. Each P_i generates a polynomial $F_i(x) = \sum_{k=0}^{t-1} f_{i,k} x^k \ mod\ (q)$ where $f_{i,0} = x_i$ and $f_{i,1}, f_{i,2}, ..., f_{i,t-1}$ are random integers in Z_q. Each P_i distributes an encryption of $d_{i,j} = F_i(j)$ to other all other trustees where $i = 1, 2, ..., w$ and $j = 1, 2, ..., i-1, i+1, i+2, ..., w$. The encryption is shared with the PKI public key of P_j and attaching this encryption to the *Key Generation Transaction*. Each P_j calculates his private key share $\sum_{i=1}^{w} d_{i,j} mod\ (q)$.

3. **Encryption:** The public key of the ElGamal cipher is publically available as:
   $y = \prod_{i=1}^{w} y_i \ mod(p)$

   This public key is included in the *Election Public Key Transaction* (Transaction 2). A vote message $m \in Z_p^*$ is encrypted into a ciphertext pair (a, b) where $a = gr\ mod(p)$, $b = my^r\ mod(p)$ and r is randomly chosen from Z_q.
4. **Distributed Decryption:** For $j = 1, 2, ..., w$, every cooperating P_j provides his part of decryption $s_j = a^{d_j}\ mod(p)$.

5. **Combination:** If a set S of at least t trustees provide correct shares, $s = \frac{b}{\prod_{P_i \in S} s_i^{u_i}} mod(p)$ can be recovered where $u_i = \prod_{P_j \in S, j \neq i} \frac{j}{j-i}$.

As discussed in the Appendix, the parameter setting of ElGamal encryption may need to be slightly adjusted according to the requirements of the different e-voting applications.

As the whole e-voting system must be end-to-end verifiable, the distributed key generation mechanism and the distributed decryption must follow suit. Each trustee needs to verify that he can sum up the shares for him to a private key share to recover an unknown but unique private key, namely cooperation of any t trustees must result in the same private key. This is called verifiable key sharing, which is necessary as none of the trustees is trusted (otherwise we can just trust a trustee and do not need threshold decryption at all) and must be prevented from cheating. Moreover, independent observers must be able to verify correctness of the distributed decryption, namely the cooperating trustees use their private key shares to correctly decrypt the votes. The verification mechanism works as follows.

1. **Public Commitment:** Each P_i publishes, as part of a *Key Generation Transaction* (Transaction 1), the commitment:
   (3) $E_{i,k} = g^{f_{i,k}} mod(p)$ for $k = 0, 1, ..., t-1$
2. **Verifiable Key Sharing:** Each P_j verifies validity of $d_{i,j}$ sent to him by P_i against an equation
   (4) $g^{d_{i,j}} = \prod_{k=0}^{t-1} E_{i,k}^{j^k} mod(p)$ for $i = 1, 2, ..., j-1, j+1, j+2, ..., w$
3. **Verifiable Decryption:** Validity of the partial decryption result s_j can be publicly proved by P_j using a ZK proof of
   (5) $log_g D_j = log_a s_j$ where $D_j = \prod_{k=0}^{t-1} \prod_{i=1}^{w} E_{i,k}^{j^k} mod(p)$
   If and only if the verification is satisfied, s_j is valid and t instances of it can reconstruct the message m.

3.2.2. Pseudonyms

A pseudonym is a mechanism for identifying a voter. It is a shortened or hashed representation of the voter's public key. The affiliated key pair is only generated for the duration of a single election. The key pair also allows for ownership of transactions enforced through a signature.

Figure 4. A voter generates their pseudonym by generating an RSA key pair, hashing the public key, and hexadecimally encoding the hash. A voter's pseudonym is defined as: $V_{pseudonym} = HEX(SHA256(V_{public-key}))$

It is important to note that the voter's pseudonym is not meant to provide full anonymity in the Protocol, rather pseudonymity or partial anonymity. The authentication and authorization services are aware of the identity of the voter and possess the information to tie it to their pseudonym.

See Mix-Networks for information on full anonymity.

3.2.3. Authentication

The function of the Authentication Service is to act as a trusted source of proof of identity for a voter. N-version programming and a threshold multi-signature scheme is used to ensure that there is no single point of trust and failure for authentication. Voter authentication requirements and standards vary wildly from jurisdiction to jurisdiction. As such, the Proof Of Vote only specifies the high-level interactions between a voter and a set of authenticators, leaving the details open to the implementers. The authentication process is as follows:

1. Before each election, all authenticators produce an RSA key pair. The public portion of this keypair is embedded in the *Election Creation Transaction* (Transaction 0).
2. Voter sends Identifying-Information, Proof-of-Identity, and their pseudonym to a set of authenticators.
3. Authenticators verify Identifying-Information against Proof-of-Identity, and send back the following:
    - Canonical Identifying-Information.
    - Voter's Pseudonym.
    - Signature on the above two items as a package.
    - Standalone Signature on Voter's Pseudonym.
4. Each authenticator logs the transaction for auditability.

Identifying-Information is whatever is required to identify the user on the Electoral Roll and varies by jurisdiction; generally this would be name and address. Canonical Identifying-Information is the same information in canonical form. Proof-of-Identity is whatever supporting information is required to support the voter's claim to their identity. Examples of possible Proof-of-Identity include:

- Drivers License.
- Social Security Number.
- Proof of Address via QR Code mail-out.
- Fingerprint Recognition.
- Facial Recognition
- Digital Signature (ex: via a cryptographically enabled ID card).
- Digital Identity Attestations (ex: Civic (www.civic.com))

Figure 5.
(a) Voter sends Pseudonym, Identifying-Information, and Proof-Of-Identity to a quorum of authenticators. Authenticators send back a signed pseudonym and a signed package comprised of the pseudonym and Canonical Identifying-Information.
(b) Voter consolidates all authentication signatures on the package containing pseudonym and Canonical Identifying-Information.
(c) Voter consolidates all authentication signatures on the standalone pseudonym.

While the voter should send all Identifying-Information to all authenticators, it is not required that all authenticators check the same Proof-of-Identity. For example, one authenticator might validate the voter's address by checking that they have scanned a QR Code that was sent to their residential address via postal mail, while another authenticator might check a digital signature created by the voter scanning a smart card ID via near field communication (NFC) on their mobile device.

After authenticating with a quorum of authenticators, the voter consolidates the signatures provided for the next step of the protocol, authorization.

3.2.4. Authorization

The Authorization Service acts as a source of a voter's eligibility to vote and determines the Ballot Style ID a voter is eligible to vote with. Ballot Style ID is expected to vary from jurisdiction to jurisdiction. Similar to authentication, N-version programming and a threshold multi-signature scheme is used to ensure that there is no single point of trust and failure for authorization. The authorization process is as follows:

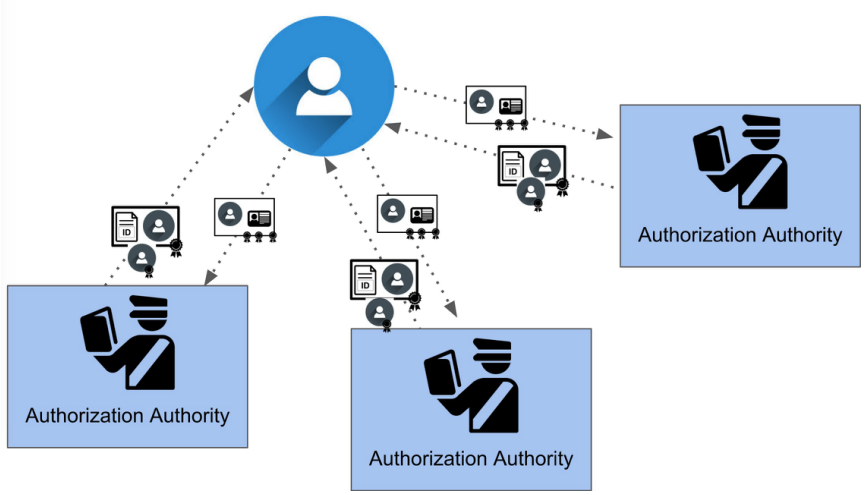

***Figure 6**. Voter sends authenticator-signed Pseudonym and Identifying-Information to a quorum of authorizers. Authorizers send back a signed pseudonym and a signed package comprised of the pseudonym and Ballot Style ID.*

1. Before each election, all authorizers produce an RSA key pair. The public portion of this keypair is embedded in the *Election Creation Transaction*.
2. Voter signs the package of Identifying-Information/Pseudonym sent to it by the authenticators, and sends it, along with its public key, to a quorum of authorizers
3. Each authorizer verifies that the public-key signs the package, and that all authenticator signatures are valid
4. Each authorizer sends back the following:
    - Ballot Style ID assigned to the voter.
    - Voter's Pseudonym.
    - Signature on the above two items as a package.
    - Standalone Signature on Voter's Pseudonym.
5. Each authorizer logs the transaction for auditability.

After authorizing with a quorum of authorizers, the voter consolidates the signatures provided for the next step of the protocol, initialization. The Authorization Service also acts as a mechanism to revoke previous authorization of voters. This could be necessary in a few cases:

1. The voter misplaces the private key affiliated with their pseudonym.
2. The voter decides to spoil their ballot, and requires a new ballot to be delivered to them.

3.2.5. Ballot Delivery

Ballot Delivery is the process by which a voter receives all information required for them to cast a valid vote. After the *Initialization Transaction* (Transaction 3) is confirmed, the voter may request a Ballot Style from a node

provisioned to deliver one. The node will examine the *Initialization Transaction* for the voter and deliver the correct Ballot Style. The Ballot Style is composed of:

1. Ballot Style ID.
2. Submission Style ID.
3. Election Public Key.
4. Marking style.
5. Submission Style.

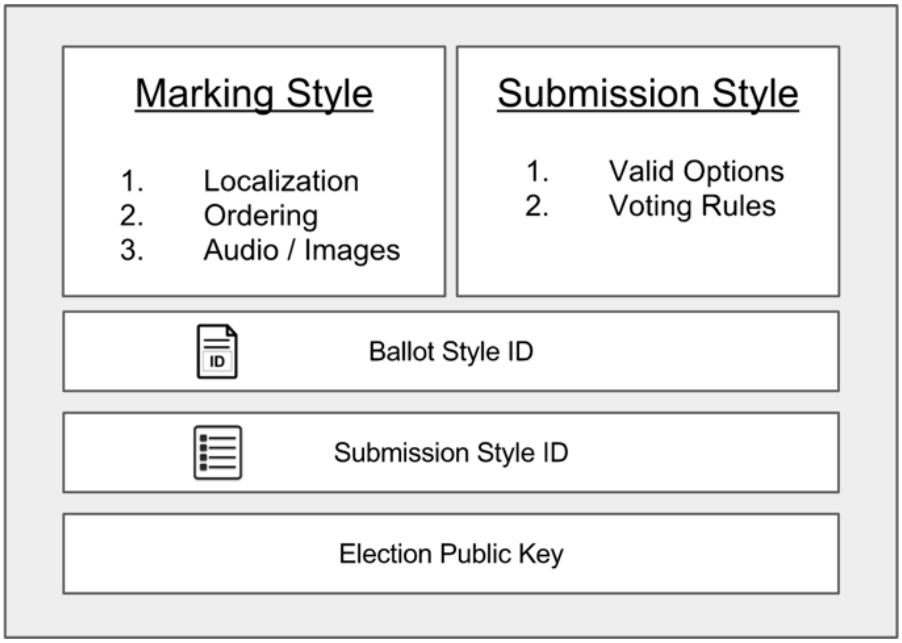

***Figure** 7. Ballot Style.*

The Ballot Style ID is the unique ID for this Ballot Style, as defined by the EMS. Submission Style ID is a unique ID for the Submission Style attached to this Ballot Style. Note that multiple Ballot Styles may all share the same Submission Style. The Marking Style is the information needed by the voter's device to display the ballot for marking. It may include text in multiple languages, images, audio for the visually impaired, and a unique ordering of choices. The Submission Style is comprised of information that is required to correctly tally the marked ballot. It might include valid choices or any other information required to confirm the validity of a marked ballot.

After receiving the Ballot Style, the voter will check the *Election Creation Transaction* to make sure the Ballot Style ID, Submission Style ID, and the hash of the Ballot Style match what was delivered. The voter is now assured that they have been delivered the correct Ballot Style.

3.2.6. Vote Encryption

Although RSA is the only public key encryption algorithm recommended by FIPS (Federal Information Processing Standards), it is not suitable for vote encryption as it does not support re-encryption and is not semantically secure.[18]

[18] Federal Information Processing Standards (FIPS) are publicly announced standards developed by the United States federal government for use in computer systems by non-military government agencies and government contractors. Similar international standards may be found under the Institute of Electrical and Electronics Engineers (IEEE), and the International Organization for Standardization (ISO).

A semantically secure encryption algorithm revealing no information about the encrypted vote is needed; it must support re-encryption to enable the re-encryption mix-network chosen for the Protocol.

The Protocol uses the most common re-encryptable and semantically secure public key encryption algorithm, ElGamal encryption, for ballot encryption. It is also the encryption algorithm most frequently employed by re-encryption mix networks. Another merit of ElGamal is that a threshold encryption algorithm with distributed key generation is easily supported such that the private key to decrypt votes can only be used with the cooperation of a threshold number of private key holding trustees. Vote encryption steps are as follows:

1. Voter's device notes encryption key provided to them during ballot delivery.
2. Voter's device concatenates their Ballot Submission Style ID and their Marked Ballot. This becomes the plaintext that is encrypted into a vote ciphertext.
3. Voter's device encrypts the plaintext v into an ElGamal pair $(a, b) = (g^r, vy^r)$ where r is randomly chosen from Z_q.
4. Voter's device produces a Range Zero Knowledge Proof that the correct Ballot Submission Style ID is prepended within the ciphertext. This can be accomplished by casting the entire plaintext to an integer and providing a ZKP that this integer is within a range. Because the Ballot Submission Style ID is prepended, it becomes the most significant digits in this integer, and a range proof serves as a ZKP that the correct Ballot Submission Style ID is present in the ciphertext.
5. The voter packages the following information to create a Vote Transaction (Transaction 5):
    a. Pseudonym of the Voter.
    b. Encrypted Vote.
    c. Range Zero Knowledge Proof.
    d. Ballot Submission Style ID.
    e. RSA signature of the above information.

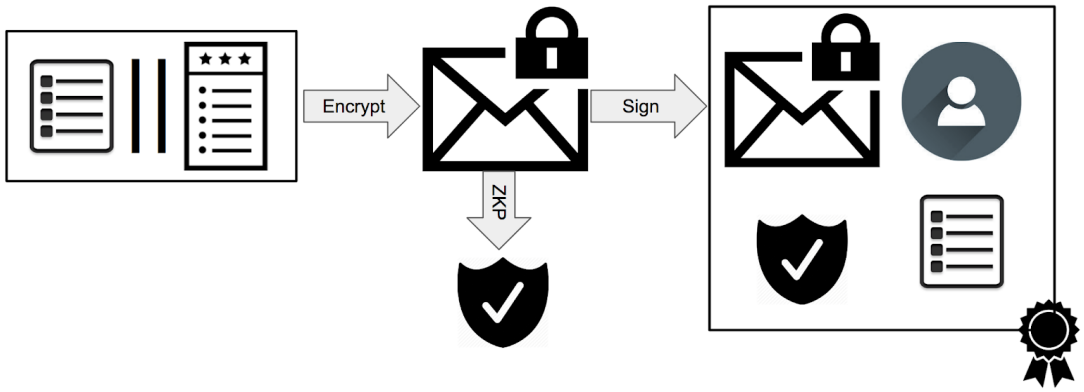

***Figure 8**. The voter concatenates their Ballot Submission Style ID and their Marked Ballot. This becomes the plaintext that is encrypted into a vote ciphertext. The voter encrypts their vote with the key provided to them during ballot delivery and produces a Zero Knowledge Proof. This information, along with their pseudonym and information obtained from the Ballot Delivery Transaction, is packaged and signed by the voter. This package becomes a Vote Transaction (Transaction 5).*

Implementation of the Ranged Zero Knowledge Proof can be found in the Appendix.

3.2.7. Interactive Device Challenges

A Benaloh Challenge[19] can be used to maintain the honesty of the voting device. In Proof of Vote's implementation, the voter uses an auditing device (for example, a cellphone) to audit a voting device (for example, a DRE machine). In principle, any device can be used to audit any other device. 3rd party vendors are encouraged to create Proof Of Vote® companion applications that can act as Benaloh Challenge auditors.

The Benaloh Challenge keeps voting devices honest by having the voting device commit to an encrypted vote before the voting device knows if that encrypted vote is going to be audited or cast. It works as follows:

Step 1: After a ballot has been marked and the vote package encrypted, but before a Vote Transaction (Transaction 5) is submitted to the blockchain, a SHA256 hash of encrypted vote and a 4 digit Auditing PIN is encoded in a QRCode and displayed on the voting device. The voter is given the option to use a auditing device to audit their encrypted vote, or cast their vote as is. See Figure 9(a).

Step 2: If the voter chooses to audit, they select that function on their auditing device and scan the QR Code. The auditing device is now in possession of a commitment (a hash of the encrypted vote), but the voting device does not yet know it is being audited. See Figure 9(a).

Step 3: The voter reads Auditing PIN displayed on their auditing device, and enters it on the voting device. This begins the audit process. See Figure 9(b).

Step 4: The voting device displays multi-frame QR Code with all information required to audit the encrypted vote. A single QR Code cannot hold enough information for the full auditing information to be transmitted from the voting device to the auditing device, so instead the voting device flashes a series of QR Codes that together contain all information — the pattern repeats to protect against the case of the auditing device missing any single frame. See Figure 9(c). The following information is transmitted:

1. The Marked Ballot.
2. The Ballot Submission Style ID.
3. Pseudonym of the Voter.
4. Range Zero Knowledge Proof.
5. The randomization factors used to encrypt the vote.

Step 5: The auditing device receives all information and reconstructs a copy of the encrypted vote. The auditing devices checks the hash of this encrypted vote against the hash displayed in the QRCode commitment at the start of the audit process. The auditing device also displays the Marked Ballot to the voter so that they may verify that their choices are correct. The voting device re-encrypts the vote with new randomization factors. See Figure 9(d).

Step 6: The vote is now ready to cast again. If the voter wishes, they may modify their Marked Ballot before submission, hiding their real choices from the auditing device. Once they have done so (or opt not to do so), they may choose to rescan the QRCode commitment from their auditing device again. They may choose to audit the encrypted vote again, or use the auditing device to check to make sure the cast ballot is properly submitted to the blockchain. See Figure 9(e).

[19] Benaloh, Josh, and Ismail Cem Paya. "System and method for non-interactive human answerable challenges." U.S. Patent No. 7,337,324. 26 Feb. 2008.

Step 7: The auditing device waits for the correct Vote Transaction (Transaction 5) to show up on the blockchain. The voter casts their vote using the voting device. See Figure 9(f)

Step 8: Both the voting device and the auditing device confirm that the correct encrypted vote is present on the blockchain. See Figure 9(g).

While it does not completely guarantee that all votes are cast as intended, the Benaloh Challenge makes it difficult (to varying degrees of difficulty pending the number of challenges) for dishonest voting applications to successfully tamper with election results. Everytime a dishonest voting app attempts to tamper with a cast vote, there is a chance that a voter will audit the tampered vote and the dishonest app will be discovered. While any single attempt to corrupt an encrypted vote might succeed by chance, widespread vote manipulation becomes practically impossible without discovery.

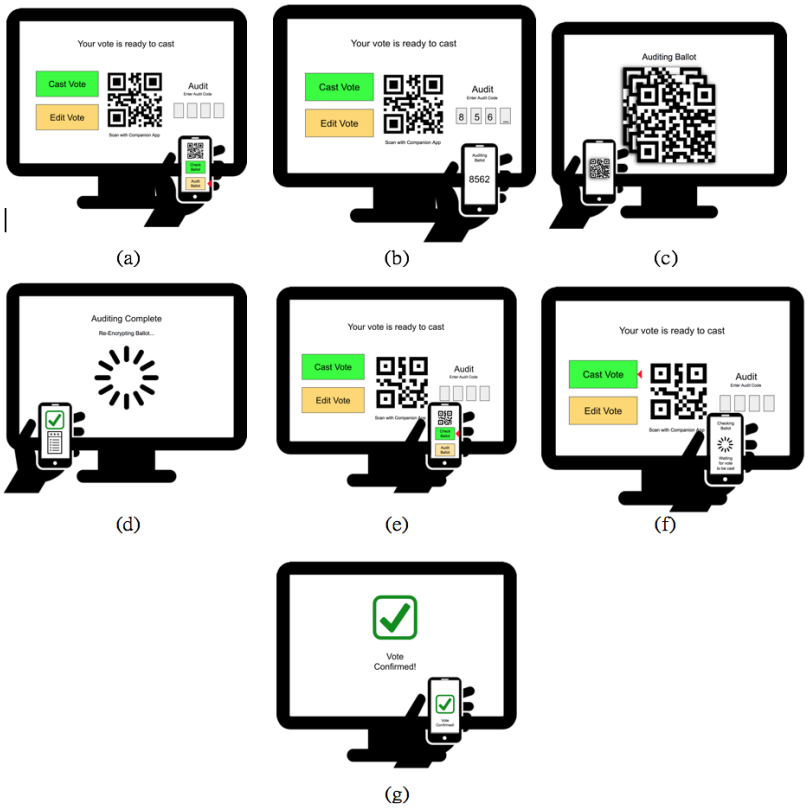

Figure 9. *Benaloh Challenge. (a) The voting device commits to the encrypted vote by displaying a QR Code of the hash. (b) The voter scans the QRCode with the auditing device. (c) The voter initializes an audit. (d) The voting device transmits all information to the auditing device. (e) The auditing device shows the marked ballot to the voter. The voting device re-encrypts the ballot. (f) The voter uses the auditing device to check the cast vote. (g) The voter casts the vote. (h) The auditing device confirms that the vote is on the blockchain.*

Decentralized Challenges

One of the long standing challenges related to human interactive proofs is how they can be simplified, embedded, automated, or abstracted such that the friction of going through the proof is mitigated so that a statistically significant number of voters actually check the system to guarantee the integrity of the system's behavior.

In order to facilitate this, Proof of Vote introduces the idea of a ***Decentralized Challenge Transaction* (TRANSACTION 5A)**. In a traditional Benaloh challenge it is possible that a compromised device can convincingly fool the voter into believing that their vote was correctly encrypted while submitting the ciphertext of a different vote; the compromised *"defeat device"*[20] would be able to deliberately accomplish this deception if it can predict with conviction which voters will or will not challenge. Even with many devices issuing Benaloh challenges, it is possible for multiple compromised devices to trick voters. In a decentralized version of the Benaloh challenge, the challenge is offered after the *Vote Transaction* is issued to the blockchain. If the voter chooses to challenge, they produce a *Decentralized Challenge Transaction*, which includes the following:

1. The randomization used when encrypting the vote
2. The plain text vote

In the case that the vote is correctly encrypted: each node would be able to verify the vote was correctly encrypted using the information provided in the *Decentralized Challenge Transaction*. If the vote is verified to be correctly encrypted, the challenged *Vote Transaction* is voided allowing the voter to cast a new secret ballot.

In the case that the vote is incorrectly encrypted: each node on the blockchain would be able to identify the vote was incorrectly encrypted using the information provided in the *Decentralized Challenge Transaction.* The public key as well as the authentication service can subsequently be used to contact the voter to inform them of a compromised device.

By enabling each node in the network to verify the correctness of encryption, *Decentralized Challenge Transactions* allow for cast-as-intended verification without reliance on voter's client device; furthermore, this shift not only relieves responsibility of voting device of the challenge, but allows for discerning if a voting device is compromised.

Lastly, decentralized challenges are more seamlessly embedded into the voting process itself allowing for statistically significant number of challenges to be automated into the election definition, for example, providing a mathematical degree of confidence that the number of compromised voting devices is within some acceptable range as determined by the Election Authority. Additionally, the design of these decentralized challenges can in theory allow for incentivizing/rewarding voters who help audit the system.

3.2.8. Vote Submission

[20] A **defeat device** historically refers to hardware, software, or design that interferes with or disables emissions controls for motor vehicles under real-world driving conditions, even if the vehicle has passed formal emissions testing. In the context of this Protocol, we've adopted the term to refer to compromised voting devices that pass traditional Benaloh challenges, but proceed to deceptively encrypt votes incorrectly. See https://en.wikipedia.org/wiki/Defeat_device for reference.

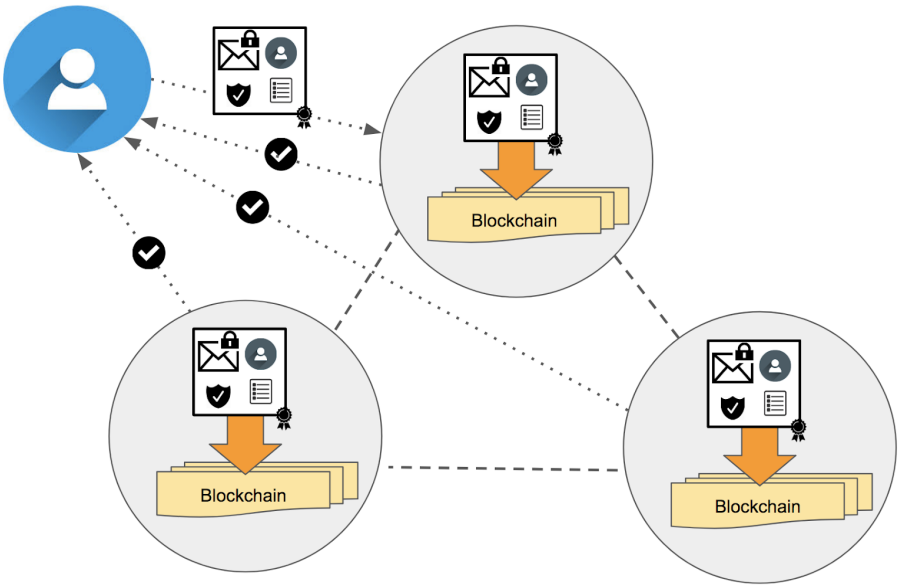

Figure 10. A voter submits their vote via a Vote Transaction (Transaction 5) to any node on the network. The node validates the submitted vote and submits it as a transaction on the blockchain. After being validated by all nodes, the transactios is written to the blockchain and the voter receives confirmation that their vote is on the blockchain.

Votes are stored on the blockchain as part of a *Vote Transaction* (Transaction 5). After validating the received *Vote Transaction*, the node packages it along with other transactions to create a block on the blockchain. The voter waits (usually only a few seconds) for a new block to be created on the blockchain with their vote. Once the voter sees the block, they can be assured that their vote is irrevocably written to the blockchain. Nodes validating the vote as part of creating the block will validate the following:

1. The given pseudonym is part of an *Initialization Transaction* (Transaction 3) and that *Initialization Transaction* has not been cancelled by a *Revocation Transaction* (Transaction 4).
2. No other *Vote Transaction* (Transaction 5) exists for this pseudonym. Note that some elections may allow for multiple *Vote Transactions* per voter, with only the last one being tallied. If this is the case, validation only requires that there be no other *Vote Transaction* for this pseudonym for the current block.
3. The Zero Knowledge Proof provided proves that the encrypted Submission Style ID matches the provided Submission Style ID on both the *Vote Transaction* and the relevant *Initialization Transaction*.
4. The signature on the *Vote Transaction* is valid and matches the public key found in the relevant *Initialization Transaction*.

3.2.9. Mix-Networks

The Proof of Vote protocol uses a combination of pseudonyms and mix networks to anonymize voters and their votes. Pseudonyms provide partial anonymization. Mix-networks (mixnets) provide full anonymization of all voters and their votes. The use of mixnets is an optional component of the Protocol, and may be elided where full vote anonymization is not needed.

An ElGamal re-encryption mixnet is used to both shuffle the votes (such that input order cannot be matched to output order) and re-encrypt the votes (such that input ciphertexts cannot be matched to output ciphertexts). The mixing nodes must produce zero knowledge proofs of correct re-encryption and shuffle — these zero knowledge

proofs are written to the blockchain as part of *Shuffle Transactions* (Transaction 7) and form part of end-to-end-verifiability.

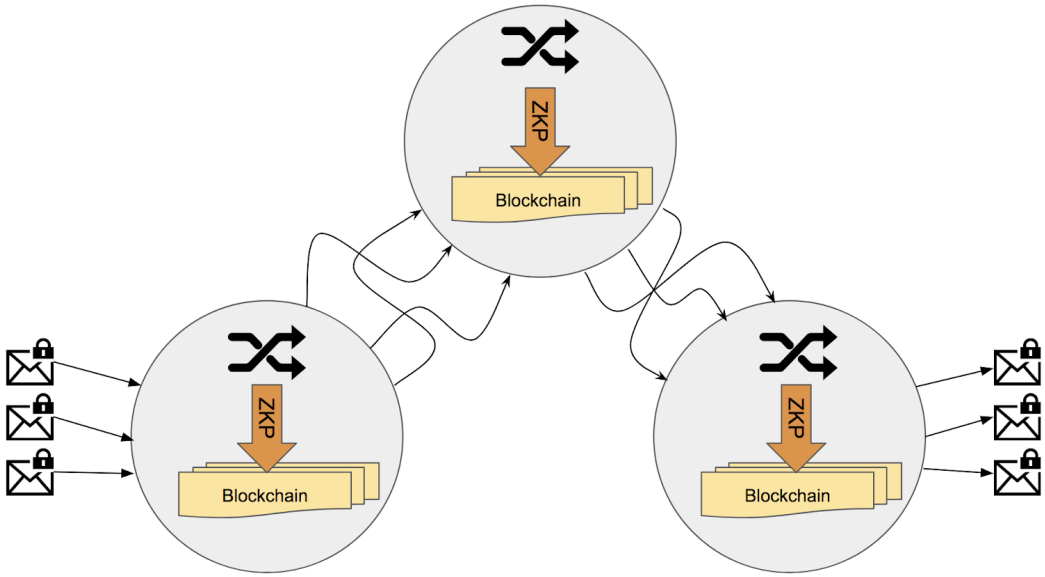

***Figure 11**. An example of a three-node-mixnet.*

Encrypted votes are passed into the first mixer node, where they are shuffled and re-encrypted before being passed onto the next node. A ZKP of valid mixing is written to the blockchain (each ZKP of valid mix mathematically guarantees that the post-shuffle votes are equivalent to the pre-shuffle votes). Each node performs this same set of operations with unique randomization factors known only to that node. The end result is a batch of encrypted votes that are fully anonymized.

A re-encryption mix network consists of several mixer nodes; each performs a shuffling to permute the input ciphertexts after re-encrypting them such that the messages contained in the output ciphertexts is a permutation of the messages contained in the input ciphertexts. The shuffling node has to publicly prove validity of its shuffling. Several mixer nodes are employed to prevent a single point of trust; while a single mix node cannot change the value of the votes themselves (due to the fact it must provide a proof of valid shuffle), it could potentially store the order of the shuffle to break the anonymity of the voter. By utilizing several mixers, we guarantee vote anonymity as long as at least one mixer is honest.

The following properties must be satisfied in a shuffling protocol:

- **Correctness:** if the shuffling node strictly follows the shuffling protocol, the shuffling protocol ends successfully and the plaintexts encrypted in the output ciphertexts are a permutation of the plaintexts encrypted in the input ciphertexts.
- **Public verifiability**: the shuffling node can publicly prove that it does not deviate from the shuffling protocol.
- **Soundness**: a successfully verified proof by a shuffling node guarantees that the plaintexts encrypted in the output ciphertexts are a permutation of the plaintexts encrypted in the input ciphertexts without any trust assumption on the shuffling node.

- **Zero knowledge (ZK) Privacy:** the permutation used by the shuffling node is not revealed. More formally, a simulated transcript indistinguishable from the shuffling transcript can be generated by a polynomial party without any knowledge of the shuffling node's secret inputs.

In this mix network, a shuffling scheme is designed based on ElGamal encryption.[21] However, as ElGamal encryption is not additive homomorphic, it is difficult to analyse any linear relation between the messages in the input ciphertexts and the messages in the output ciphertexts resulting in a more challenging soundness analysis. Under a formal privacy model, zero knowledge of the shuffling transcript can be formally proved.

See Appendix: Mix-Networks for mixnet scheme details.

3.2.10. Vote Decryption

Vote decryption happens after all votes are anonymized via the mixnet in a two step process:

1. Partial Decryption: For every vote, each trustee node produces a *Partial Decryption Transaction* (Transaction 8). This transaction contains a partial decryption of the ElGamal encrypted vote, along with a Zero Knowledge Proof of correct decryption. For the transaction to be valid, the Zero Knowledge Proof of correct decryption must be valid.
2. Full Decryption: Once a sufficient number of trustees have produced a *Partial Decryption Transaction* for a given vote, any node may produce a *Vote Decryption Transaction* (Transaction 9), which combines a set of partial decryptions into a full decryption. The *Vote Decryption Transaction* contains the fully decrypted plaintext for the vote.

3.2.11. Election Finalization

After the votes are decrypted, an *Election Finalization Transaction* (Transaction 10) is produced and votes are ready to be tallied. This transaction contains a hash digest of all transactions that make up the election, and a hash digest of all decrypted votes.

The purpose of this transaction is to allow off-chain distribution of the election for end-to-end verification and tally, while still tying these off-chain representations to the blockchain. For example, an election authority may publish the hash digests from this transaction along with the official results. Members of the public wishing to verify an election might download a package from the election authority's website containing all transactions, verify it's veracity against both the digest published with the official results and the *Election Finalization Transaction,* and perform an end-to-end verification of the election outside of the context of a blockchain. This may have particular importance in the future, where some may wish to verify an election stored in an archive long after the blockchain network has shut down.

4. Challenges and Concerns

The following section details real and common concerns that academics, public interest groups, election officials, and voters alike have with voting generally and specifically within the context of the Protocol. The subsequent

[21] Note that ElGamal encryption is much more efficient than Paillier encryption when the same level of security is required.

discussion of each concern is rooted in pragmatism and a contextual understanding of the procedural and organizational support structures in place for any consequential election in addition to these technical measures.

Historically, arguments have been made by academics and electoral interest groups that any imaginable risk of even immaterial nature is unacceptable; in other words – a perfect voting system. In practice, this proposed standard to which any voting schema is held is not applied to any other human affair and most importantly, not even currently implemented supervised and paper-based voting systems would pass these standards.

The fact of the matter is, no system, paper or electronic is without the risk of compromise.

In the formation of a technically sound voting protocol, there exists an optimum balance between the underlying security of the scheme and the viability and complexity of its practical implementation. This Protocol optimizes for both security and efficacy which we believe exceeds that of traditional methodologies like paper-based systems.

General Overview of System Requirements and Concerns

Category	Concern	Proof of Vote Protocol Consideration
Voter Anonymity	How does the protocol maintain unprovability of voting and the inability to link a voter to a vote?	Voter anonymity is achieved via a re-encryption mix-network. After votes are mixed, there is no link between a voter and a vote, given at least one honest mixer.
Voter Anonymity	How does the protocol address collusion of key holders?	The protocol does not address this issue. Trustees should be selected so as to represent a wide variety of parties with heterogeneous interests and motivations. If all trustees are located within a single jurisdiction, a court order may compel all trustees to reveal their key and deanonymize voters if an election is ongoing. Because trustees destroy their key upon election finalization, this is not possible retroactively.
Voter Anonymity	How does the protocol maintain forward election integrity?	Partial forward election integrity can be achieved by trustees destroying their private election keys upon election finalization. However, fundamental advances in cryptography or computing that is able to recover an ElGamal private key from it's public portion or ciphertext will be able to de-anonymize voters retroactively. *See Quantum Computing Considerations for more analysis.*
Voter Authentication	How does the protocol ensure that only eligible voters cast votes?	Proof of Vote guarantees this as described by the Authentication Process.
Voter Authentication	How does the protocol handle remote voter identification ensuring that a voter is who she says she	The Protocol is flexible to accommodate multiple authentication methods. Further commentary on remote authentication can be found here.

	is in the absence of her physical presence?	
Voter Authentication	How does the protocol maintain “One Person, One Vote” or more generally that no voter can vote more than allowed as defined by election rules?	Proof of Vote guarantees this through use of pseudonyms and election definitions which define the permissible number of votes per eligible voter. Each *Vote Transaction* must be unique for the given pseudonym to be a valid transaction.
E2EVV	How can voters achieve usable Cast as Intended verification?	Cast as Intended is achieved by use of a cast-or-challenge Benaloh Challenge.
E2EVV	How does the protocol ensure a statistically sufficient number of verification checks process to guarantee effectiveness?	Every transaction is checked by every node. The protocol therefore has 100% coverage on all verification checks.
E2EVV	How can voters be sure that their vote was counted as cast?	By verifying that their encrypted vote is on the blockchain, that all shuffle and decryption transaction are correct, and by independently verifying the tally, a voter can be sure that their vote was counted as cast.
E2EVV	How can voters be sure that every collected vote was counted as cast?	Guaranteed by Proof of Vote design - detailed in End-To-End Verification.
E2EVV	Does the protocol mandate reliance on a secondary system to do verification checks? Why should a voter trust a verification tool?	Most voters will opt to trust blockchain nodes, but technically competent voters may opt to download the entire transaction set and do a full end-to-end verification, using their own tools if desired. This is further discussed in Decentralized Challenges.
Transparency of Software	How publicly open and accessible are the voting software, processes and mechanisms that dictate an implementation of this protocol?	The Protocol itself is fully open to the public and available for critique, review, and feedback. Any implementations of the protocol should be made fully available for audit, review, and certification.
Software Independence	If an error occurs in the voting software, how is it guaranteed that this will not cause an undetectable error in the election outcome?	Use of blockchain ensures that any software errors that occur are caught by other blockchain participants and are not realized as erroneous transactions on the blockchain. Over the long term, N-version programming will be used to ensure that not all blockchain nodes are running the same implementation. Heterogenous implementations of the same protocol ensures protocol robustness and limits implementation bugs to the subset of nodes running that particular implementation.

Operability	How does the protocol ensure the software does what it is supposed to do?	By having a large number of participating nodes running a heterogeneous implementations of the protocol, correct software implementation of the protocol is guaranteed. On a voter's device, a Benaloh challenge is used to guarantee software correctness.
Operability	How does the protocol ensure that voters receive only the proper authorized ballot style?	Proof of Vote guarantees this as described by the Authorization Process and the Ballot Delivery Process.
Operability	How does the protocol prevent falsification of votes (by both insiders and outsiders)? How does the protocol ensure only valid votes are tabulated? How is ballot box manipulation prevented?	Because every *Vote Transaction* is verified by every node on the blockchain, there is no "insider" that is able to unilaterally modify the contents of the "ballot box" (blockchain).
Operability	How do recounts work?	Because votes are decrypted post re-encryption mix-shuffling prior to tabulation, tallying of decrypted votes is repeatable indefinitely.
Interoperability	How does the protocol work within the context of offline elections? How does it integrate with other mediums of participation?	Because the output of the protocol is a durable enumeration of anonymized decrypted plaintext votes, they could in practice be tallied and combined with any other set of anonymized plaintext votes, regardless of the medium of participation.
Dispute Resolution	If voters (or observers) detect that the system faltered, how can they prove it such that an independent third party can correctly solve the dispute?	Ultimately, protocol dispute resolution mechanisms should hold applications of the Protocol accountable, discerning between real identification of attacks and false claims of identifications to nullify any doubt cast by false claims on the integrity or accuracy of election results. Traditional paper-based voting has proven particularly susceptible to false claims and resulting doubt, despite the existence of paper trails, given there is generally no practical opportunity for voters to ever unearth errors and trigger dispute resolution mechanisms. In contrast, the transparency and verifiability of the Protocol produces evidence (mathematical proofs of correct operations immutably transcribed on an election's blockchain following consensus of authorized nodes) of maleficence or proof of proper behavior in most conceivable scenarios, affording relative ease of dispute resolution by authorized nodes potentially composed of election authorities, election observation and monitoring organizations, and auditors.
Dispute Resolution	What procedures are in place to ensure that **credible claims** of error	Proof of Vote does not cover procedural recourse for credible claims of error, as these legal considerations are determined by law and regulation presiding over an election.

	are acted on appropriately?	However, Proof of Vote does allow for the identification of credible claims of error.
Dispute Resolution	How can the system (or the authorities) defend against **false claims** that the system misbehaved?	Every transaction that has been written to the blockchain has been confirmed as valid as determined by consensus built against the validity requirements for what classifies valid or malicious activity (these requirements are discussed here). Furthermore, zero knowledge proofs of correctness tied to different transactions within Proof of Vote (digital key signature generation, re-encryption shuffles, partial decryption, proper ballot submission style ID, range proof of vote submission within permutation of possibilities) prove the integrity of the transpired transactions. False claims of system misbehavior can be demonstrably discredited by Proof of Vote design. The component of the Protocol that lacks this absolute certainty relates to cast-as-intended checks or checks of proper encryption because these checks demonstrate (to a statistically significant degree of confidence) that a client device is properly encrypting a vote, but cannot show a voter their vote (to maintain coercion resistance).
Collection Accountability	How can voters who detect that their vote has not been collected provide convincing evidence to election authorities that their vote has not been collected?	It is trivial for a voter to show that a *Vote Transaction* with their pseudonym is not present on the blockchain. Voters may query any number of blockchain nodes to check that all nodes have confirmed that the voter's *Vote Transaction* is present on the blockchain.
Auditability	How does the system ensure reliable, unforgeable and unchangeable voter records for auditing purposes that are as effective as paper ballots (without sacrificing ballot secrecy)?	Auditable anonymized decrypted plaintext votes are guaranteed reliable, unforgeable, and unchangeable by the Proof of Vote blockchains's indelibility indefinitely. This property more generally holds for all transactions written to the blockchain over the course of a given election, not just to vote transactions.

4.1. Voter Anonymity

In IT systems, there are usually two ways to protect user privacy. Primarily, if a user's data and operations are confidential (e.g. through encryption), their transactions are secret and their privacy is protected. Alternatively, if their data and transactions need to be public, they can remain anonymous so that those transactions are not known to belong to them. These two types of user privacy are referred to as *absolute privacy* and *relative privacy* respectively.

In both blockchains and e-voting, anonymity is employed to guarantee user privacy. In blockchain, as all the transactions are open and public to be verified by all the nodes, user privacy cannot be guaranteed by data

confidentiality. As absolute privacy is impossible, the users keep themselves anonymous to achieve relative privacy. As their transactions cannot be linked to their real identities, the users' privacy is guaranteed. As e-voting schemes usually achieve end-to-end public verifiability, all the votes must be opened and tallied publicly.

Except for some voting schemes employing special tallying methods like homomorphic tallying, most e-voting schemes decrypt the ballots separately and tally them one-by-one to support a flexible vote format. With the contents of all the votes published, absolute privacy has already failed and the only alternative is relative privacy. So, the votes must be counted anonymously without any link to the corresponding voters.

In e-voting, a voter's anonymity only lies in the unlinkability between her and her vote and does not require that she must always remain anonymous. In contrast, every voter must be identified before casting a vote, usually by an election authority or authenticator to ensure that they have the right to vote. A voter's pseudonym can be regarded as a token or certificate issued by the authenticator (e.g. in the form of the authenticator's digital signature) and used by the voter to prove their qualification for voting. The pseudonym enables the voter to vote pseudo-anonymously while being tied to be a valid voter. However, because the pseudonym is signed by the authorizer, the authorizer can recover the voter's identity from their pseudonym, especially in a publicly verifiable e-voting scheme where all the pseudonyms are used publicly.

Blind signatures may be employed to to make the identity behind a pseudonym unrecoverable. However, blind signatures have several drawbacks in that a voter's interaction with an authorizer leaks information about the voter. The IP address associated with the voter may be visible to an authorizer, and correlations in the time that a voter obtains a blind signature from the authorizer and the time they cast their vote may unmask a voter.

Since blind signatures have concerning side effects; anonymous routing is a more reliable and robust tool to maintain anonymity. No matter whether masked by pseudonyms, if all the votes are submitted through an anonymous communication network which guarantees unlinkability between its inputs (namely the submitted votes which may reveal the voters' identities although being encrypted) and its outputs (namely the counted votes which are decrypted and published) and is immune to traffic analysis, relative privacy of the voters is achieved.

The Protocol's implementation of mix-networks satisfies such a traffic-analysis resistant anonymous communication network providing unlinkability. It employs multiple mixers (routers), each routing and shuffling all the input ciphertexts (encrypted votes in the case of e-voting) using a random permutation in turn. If at least one mixer conceals its permutation, no output of the mix-network can be traced back to any of the inputs. As all the ciphertexts are routed in large batches and mechanisms like re-encryption and partial decryption are employed by the mixers to randomize the ciphertexts it shuffles, traffic analysis is fully mitigated in a mix-network.

Mix-networks achieve the privacy requirements that self-selected pseudonyms, authority-certified pseudonyms and blind signatures cannot guarantee. Consequently, employing a mix-network is inevitable to achieve voters' privacy in any e-voting scheme except those employing homomorphic tallying, which strictly limits the range of applications and strengthens the dependency on strict vote format needed for verification.

Other techniques intended to protect anonymity and privacy are discussed in the Appendix.

4.2. Voter Coercion

All voting media are susceptible to voter coercion. Online voting is no different in this regard. However, before delving deeper into a discussion on the effects of online voting on voter interference, it's important to understand the

societal measures that lead to voter interference. Ignoring these contextual dimensions allows for baseless arguments that internet voting affords more opportunities for voter interference than traditional mediums like paper voting, polling place voting, and postal voting.

This is because online voting has historically been treated with adversarial and unsympathetic skepticism and consequently held to a higher theoretical standard than other media of voting:

> 'people expect much more from electronic voting schemes than from paper based systems ...'[22]

> 'Internet based voting does not introduce these problems [vote buying and coercion], but it does have the potential to exacerbate them by extending the reach and data collection abilities of an attacker' [23]

> 'voter coercion and vote buying ... are highly scalable in an electronic environment' [24] [25] [26] [27]

However, empirically, it's been shown that "available opportunities for voter interference vary primarily not with technological safeguards, but with cultural, political, and economic conditions that govern interactions between people within a particular society" meaning it's less a factor of the medium of delivery and more a factor of encompassing mores of a given society.[28] The very same is true for the relationship between the secret ballot and voter interference; to say that this one instrument is responsible for the elimination of voter coercion at scale is to ignore a variety of encompassing procedural and societal factors that also reduced the opportunity for such interference.

A country's strength and independence of electoral administration, general level of corruption, egalitarian versus hierarchical society, and state of civil society are all more indicative of the extent of voter interference than the medium through which electors vote.[29] When every medium of voting is stripped of this social context, the media are similarly prone to voter interference and the opportunities for coercion and bribery are similarly severe. Voter interference in the context of online voting is nuanced and a real concern, but should be practically considered to address the successful ways that coercion can be resisted.

What makes voter coercion such a difficult problem in the context of e-voting, is that both public verifiability and receipt-freeness must be achieved. The former is required by the public end-to-end verifiability property widely desired in secure e-voting. The latter is necessary to prevent vote coercion and vote selling, which are feasible once a voter has a receipt to show their vote selections. Unfortunately, very often these two security properties conflict, one requiring accountability of all the votes while the other prohibiting veracity of any vote. Proof of Vote attempts to balance the two security properties.

---

[22] Chevallier-Mames, B., Fouque, PYA., Pointcheval, D., Stern, J. and Traoré, J. (2010) 'On Some Incompatible Properties of Voting Schemes', in D. Chaum, M. Jakobsson, R. Rivest, P. Ryan, J. Benolah, M. Kutylowski and B. Adida (eds), Towards Trustworthy Elections, Heidelberg, Springer.

[23] Juels, A., Catalono, D. and Jakobsson, M. (2010) 'Coercion - Resistant Electronic Elections', in D. Chaum, M. Jakobsson, R. Rivest, P. Ryan, J. Benolah, M. Kutylowski and B. Adida (eds), Towards Trustworthy Elections, Heidelberg, Springer.

[24] Spycher, O., Koenig, R., Haenni, R. and Schläpfer (2012) 'A New Approach towards Coercion - Resistant Remote E-Voting in Linear Time', in G. Danezis (ed.), Financial Cryptography and Data Security, Heidelberg, Springer.

[25] Joaquim, R., Ribeiro, C. and Ferreira, P. (2010) 'Improving Remote Voting Security with Code Voting', in D. Chaum, M. Jakobsson, R. Rivest, P. Ryan, J. Benolah, M. Kutylowski and B. Adida (eds), Towards Trustworthy Elections, Heidelberg, Springer.

[26] Clark, J. and Hengartner, U. (2012) 'Selections: Internet Voting with Over the Shoulder Coercion Resistance', in G. Danezis (ed.), Financial Cryptography and Data Security, Heidelberg, Springer.

[27] Esteve, J., Goldsmith, B. and Turner, J. (2012) International Experience with EI Voting, Washington, International Foundation for Electoral Systems.

[28] Smith, Rodney (2006) Internet Voting and Voter Interference - A report prepared for the New South Wales Electoral Commission, University of Sydney.

[29] Ibid, 24.

Susceptibility to Coercion

Proof of Vote is resistant to coercion, meaning there is a mechanism by which an affected voter can proceed to vote without the ability for a coercer to discern whether or not they have been successful in their coercion. There are however some caveat susceptibilities. The use of re-encryption mixnets and pseudonyms ensures that votes are fully divorced from real identities, which prevents the voter from being coerced after their vote has been cast. Despite these protections, a naive implementation of the Protocol is still susceptible to two types of coercion:

1. over-the-shoulder coercion and,
2. the Italian Attack.

We address each of these types of coercion in turn, and discuss how a mature Proof of Vote implementation operating in a cooperative jurisdiction can mitigate these types of coercions.

1. Over-The-Shoulder Coercion

Clark et al show that allowing a voter to vote multiple times can be effective in preventing over-the-shoulder coercion.[30] In this model, only the last vote is counted towards the tally and, if a voter is over-the-shoulder coerced, they simply vote again a second time in private. This is not a complete solution, since a voter may be coerced just before polling is closed, not allowing them to vote again. However, it mitigates the ability of a coercer to operate by significantly timeboxing when the coercion must happen.

Implementing over-the-shoulder coercion resistance presents several challenges:

1. Most jurisdictions do not allow voters to vote multiple times. Jurisdictional statutes would need to be adjusted to allow voters to vote multiple times in order to recast their vote and mitigate over-the-shoulder coercion.
2. The Protocol can be adjusted such that votes are fully secret from the public until after the mix-net fully anonymizes the votes. Only permissioned nodes would be able to view cast ballots on the blockchain, and voters would need to trust the consensus of nodes that the votes are valid. Nodes would need to be prevented via governance from colluding with over-the-shoulder coercers.
3. Given the above challenges and caveats, Proof of Vote can be adjusted to mitigate over-the-shoulder coercion, provided jurisdictional support and acceptance of reduced verifiability implications.

2. Italian Attack

The Italian Attack, described by Cosmo[31], is also known as a ballot-as-signature attack. In this attack a coercer (usually in the form of vote buying or other post-tally coercion methods) asks the voter to mark their ballot in such a way that it is unique. By making the ballot unique (or statistically improbable), the coercer can search the public

[30] J Clark and Hengartner - U Cryptography, "Selections: Internet Voting with Over-the-Shoulder Coercion-Resistance.," Financial Cryptography (2011), http://link.springer.com/content/pdf/10.1007/978-3-642-27576-0.pdf#page=58.
[31] Roberto Di Cosmo. On privacy and anonymity in electronic and non electronic voting: the ballot-as-signature attack. Hyper Articles en Ligne hal-00142440 (2), 2007.

blockchain of decrypted votes, find the voter's unique unencrypted ballot, and check to see if the voter has voted "correctly". There are two primary methods by which a ballot can be made unique:

1. Multi-contest ballot with write-ins.
2. Ranked choice ballot with many candidates.

For multi-contest ballot with write-ins, the coercer directs the voter on how to vote on all contests except one. In the one excepted contest, the coercer directs the voter to write-in a globally unique candidate which uniquely identifies the voter's ballot. Proof of Vote can prevent against this version of the Italian Attack by having the voter uniquely encrypt all contests separately, and processing each contest as a fully independent ballot. This requires jurisdictional support, since some jurisdictions require unencrypted multi-contest ballots to be presented for tally as one unified ballot. This jurisdictional requirement would need to be relaxed to allow each contest to be encrypted and unencrypted independently.

In a ranked choice voting situation with many candidates, the coercer directs the voter to vote for the candidate of the coercers choice as their first choice and then directs the voter to vote for the remaining large number of candidates in a very specific rank order. The specific rank order of candidates uniquely identifies the ballot, allowing coercion. The Protocol cannot protect against this type of attack. However, a jurisdiction can mitigate this type of attack by requiring all candidates to submit a deposit to be listed on the ballot, which is refunded if the candidate receives a threshold of votes. While the Protocol cannot prevent ranked-choice many-candidate Italian Attacks, publishing unencrypted votes to a public blockchain allows for the detection of this attack via statistical analysis, aiding in investigation and prosecution of coercers.

4.3. Voter Authentication

Proof of Vote's pseudonym implementation and blockchain ensure that authenticated voters cannot "double spend" or vote more than the prescribed number of times for a given election.

Furthermore, remote voter authentication in uncontrolled environments is already practiced at scale today. For example, in the United States, the total number of voters who vote absentee or by mail more than doubled from 12.1% in 2004 to more than ~24% in 2016 representing 33 million people.[32] That represents an incredibly substantial portion of US voters that is already navigating these "uncharted" waters. The very same standard of authentication applies uniformly for remote environments within the mandates of a jurisdiction's laws and regulations.

From a technology standpoint, as described in the Authentication Section, there is a lot more that could be done to implement stronger authentication from multi-factor authentication to biometrics, but it's ultimately at the discretion of election authorities to determine eligibility and the qualifications for authentication.

4.4. Node Collusion

Node collusion is a risk in several different contexts.

[32] Election Assistance Commission, https://www.eac.gov/documents/2017/10/17/eavs-deep-dive-early-absentee-and-mail-voting-data-statutory-overview/

Trustee and Mixer collusion presents a risk to voter anonymity. Because the trustees jointly hold the decryption key, if a sufficient number of nodes collude (or are compromised), they can decrypt votes before they are anonymized via the mixnet. Trustee collusion also presents risks to election completion. If a sufficient number of trustees collude (or are compromised), they can refuse to decrypt the votes, removing the ability for election authorities to tally election results. Mixer collusion also presents a risk to voter anonymity. If a sufficient number of mixer nodes collude (or are compromised) they could collectively break voter anonymity.

Because of the implications of trustee or mixer collusion, trustee and mixer nodes operate in a high-governance environment. Special care is taken to ensure these nodes are likely to remain impartial, and that their security posture is appropriate to the task at hand.

See Voter Anonymity for more information on related challenges and concerns.

Collusion amongst nodes participating in blockchain consensus also presents risk. Proof of Vote makes use of a Byzantine Fault Tolerant consensus algorithm. Because of the nature of this consensus algorithm[33], the entire protocol may undergo a Byzantine Failure[34] condition if more than one-third of all nodes collude (or are compromised by the same malicious actor.) Such a situation would result in transactions being dropped and the protocol rendered inoperable. However, such collusion is very visible, and with correct governance would result in detection, investigation, and ultimately banishment of misbehaving nodes. This risk can be further mitigated by having heterogenous voting power amongst consensus nodes. An election authority may choose to assign consensus voting power based on governance and certification levels that the participating node has undergone. This allows an election authority to open participation to a wide number of nodes, but also manage collusion risk in a fine-grained manner.

4.5. Endpoint Security

Bringing any system online increases the likelihood and ease of attack. Anything can be “hacked”; if governments, the world's leading cybersecurity firms, financial institutions and other consequential and high security organizations have been hacked, what can be done to secure the integrity of the voting process in the face of the numerous and severe threats?

While the Protocol uses blockchain to ensure that if some nodes are compromised the voting process as a whole remains unaffected, extreme care must still be taken in securing all nodes participating in the blockchain. If a sufficient number of nodes are compromised by the same malicious actor, these compromised nodes become colluding nodes with all the associated risks. This risk can be mitigated by having a large number of participating nodes running heterogenous implementations of the same protocol, but extreme care must still be taken to secure all blockchain nodes.

Proof of Vote treats security as a shared responsibility between the client applications running on a voter’s device, and the blockchain nodes with which they interact (as demonstrated, for example, by Decentralized Challenges). Each technology and interaction between parties must be protected against bad actors ranging from less experienced hackers (“script kiddies”) to sophisticated state-sponsored attackers. In order to properly secure the electronic voting

[33] Buchman, Ethan. Tendermint: Byzantine fault tolerance in the age of blockchains. Diss. 2016.

[34] Castro et al. "Practical Byzantine fault tolerance and proactive recovery." ACM Transactions on Computer Systems 20.4 (2002): 398-461.

process, we identify that a deployed system must address Open Web Application Security Project (OWASP) (https://www.owasp.org/), which regularly publishes updates to its web application and mobile top-10 risks meant to keep abreast threats to critical software-enabled infrastructure, and other internet threats:

1. Safeguarding Client-Server Communication
2. Protecting Against Unauthorized Node Usage
3. Auditing API usage
4. Safeguarding Client-Server Communication & Imposter Clients
5. Denial of Service
6. Other Considerations

4.5.1. Client-Server Communication

Modern encryption of communication between a client and server is the most basic start to securing the voting platform. Applications interacting with a web service or blockchain node must communicate over HTTPS using at least Transport Layer Security (TLS) 1.2 with Certificate Pinning. In addition to using server authentication to prove to a client that a server is secure and is the correct web service, TLS 1.2 must be configured to take advantage of client authentication. Client authentication requires the remote application to present its own valid certificate issued by a trusted CA. This client certificate must not be stored in an application bundle, but rather generated at the time of application installation. If the certificate is not provided or is invalid, the connection attempt must fail and terminate. [35]

The TLS connection must prefer a cipher suite which has been determined to provide forward secrecy. TLS-ECDH-WITH-AES-128-GCM-SHA256 is currently the recommended secure cipher suite which client applications and web application servers should use when communicating.[36]

4.5.2. Unauthorized Server API Usage

Beyond ensuring communication between client and web services or blockchain nodes cannot be eavesdropped upon, the application stack must make every effort to ensure that APIs cannot be utilized openly without proper client authorization. A number of steps are required and recommended in order to help facilitate trusting relationships between client and server applications:

1. Client applications must only be granted access after authenticating with the web service using the OAuth 2.0.
2. The server must maintain a record of client applications, their metadata, and their associated client id and client secret.
3. Client applications should follow authorization best practices described by the OAuth Working Group.[37]
4. Client applications and the web service should agree on utilizing PKCE as a means of ensuring OAuth redirects. PKCE prevents against URL scheme "squatting" — an attack vector which could allow a malicious application to intercept OAuth access tokens. [38]

[35] T. Dierks and E. Rescorla, "RFC-5246: the transport layer security (TLS) protocol, version 1.2," p. 7.4.6, 2008.

[36] Y. Sheffer, R. Holz, and S. P, "RFC-7525: recommendations for secure use of transport layer security (TLS) and datagram transport layer security (DTLS)," p. section 4.2, 2015.

[37] A. Parecki, "OAuth 2.0 servers: Mobile and native apps," 2017.

[38] J. Bradley and N. Agarwal, "RFC-7636: proof key for code exchange by OAuth public clients," 2015.

The inclusion of the OAuth 2.0 standard authorization procedure helps to mitigate the risk of an application redirecting a voter to a malicious site – as the URL opened in the browser can be cross-examined by a voter. If the URL does not match, the voter will know something is awry. By requiring applications to fully authenticate and be authorized by the OAuth protocol, API calls surfaced by web service are inaccessible until the client is fully vetted.

The extra layer of security added by PKCE, which helps mitigate the risk of an application squatting on a URL scheme (e.g. *com.votem.mobilevotingclient20181106://*) by introducing shared secret validation into the authorization procedure. Even if an access token request is performed by a malicious application using a valid authorization code, that application will have no knowledge of the shared secret required to retrieve an access token, causing the authorization to fail.

With proper implementation of the most recent OAuth protocols and extensions, as well as full TLS client authentication, security is spread across multiple validation mechanisms — distributing the risk across the application infrastructure. Any sufficiently suspicious application could have its client certificate revoked, or its OAuth access token revoked permanently.

Lastly, after successful authorization with the server, all requests made to the API must be signed in order to validate the identity of the requester. This goes even further to protect data as it's transmitted, by preventing parameters of API calls from being modified while in flight. Request timestamps must be provided as well. Requests must reach the web service API within a short period of time relative to the timestamp published in the API request. If the timestamp exceeds the allotted threshold, then the request must be denied by the web service.

4.5.3. API Usage Audit

All major events should provide enough information to create an audit trail of actions performed against the API. This includes OAuth authentication tokens generation and issuance, access token issuance, user authentication attempts and the IP addresses from which all data is captured. Anomalies should be identified by the system and logged by the audit system. Any detection of client application tampering, OAuth failures, or third-party verification failures, should all be reported by the web application and blockchain node where it can be presented to election officials in a way that is easily readable and searchable.

4.5.4. Application Counterfeiting & Imposter Clients

Election jurisdictions may provide official software options. Some of these methods could include sending official correspondence via mail; providing links to official software from state government websites; and providing links to official software from a trusted place of governance (city hall, polling places, offices of the Secretary of State). Correspondence to voters would contain a link that is easily recognized and reached from a web browser; a QR code that can direct voters to the correct application; information disclosing the risk of fraudulent software; information about viewing a cast vote record on the blockchain and where that can be done.

By engaging in formal, transparent correspondence with voters, forgery risk is spread across modalities. In addition to taking on illicit software forgery, a malicious party would also be required to falsify government documents.

Official ballot marking applications must be published to well-known, trusted App Stores. In the case of Apple's App Store, creation of an account requires disclosing of personal information. This developer information is verified by Apple prior to that entity obtaining the right to publish an application on the App Store. In providing information that is "valid enough", an attacker is potentially opening themselves up to significant risk to being identified.

As an example, all applications available in the Apple App Store and the Google Play Store are hosted in the United States. As a result, any application which utilizes sufficiently complex cryptography must meet U.S. Export Compliance criteria.[39] This requires software developers to register their product with the BIS to achieve proper compliance. A malicious entity would open itself to high levels of risk in achieving compliance in order to create a forged voting application intended for distribution. All applications submitted to the App Store must undergo an encryption review.[40] Similarly, Google also requires applications to declare export compliance documentation.[41]

4.5.5. Denial of Service

The risk of Denial of Service (DoS) attacks, which preclude availability to voters for a set period of time, will likely increase as e-voting achieves greater adoption. However, the risk of DoS attacks are easily mitigated even as e-voting emerges as a prevailing medium of participation in the democratic process.

In the scenario of a successful DoS attack, the underlying interoperability of the protocol means that permitted alternate methods of voting - in person, paper ballots, mail-in postal ballots, provisional - will still be available to eligible voters. This is no different from how jurisdictions offer alternate media of participation in the face of a natural disaster that renders a physical polling place unreachable. Similarly, the length of an election will also serve as a great barrier to the prolonged success of the DoS attack. Early voting provisions (as few as one week in Estonia to one month in some States in the US) will typically provide sufficient protection by limiting the temporal scope and viability of these sustained attacks.[42]

Furthermore, Proof of Votes' blockchain implementation allows for distribution of service across as many providers and networks as there are participating nodes, each of which can have their own DoS protections, which disrupts the risk of a centralized DoS attack

Ultimately, both the likelihood of a successful DoS attack and the impact of a successful DoS attack can be sufficiently mitigated in practice.

4.5.6. Other Considerations

Misconfiguration: Misconfiguration is a major risk within the context of the Protocol as proper configuration is what guarantees the security and integrity properties of the Protocol. While the Protocol is flexible in its accommodation of malicious nodes, because of the implications of full collusion, special care in a high-governance environment should be taken during configuration of trustee nodes.

Binary Modification & Runtime Alteration : If an attacker was able to successfully intercept and modify the delivery of a mobile application binary (an Android .apk file or an iOS .ipa file), code could be modified in such a way that ballot or voter secrecy could be compromised. Fortunately, iOS provides security mechanisms through its code signing and provisioning system which prevent altering of source code without resigning the binary. A client application that is modified will fail to install successfully if the contents of the binary to not match that which was

[39] "Encryption items, code of federal regulations," 2017. Title 15, Subtitle B, Chapter VII, Subchapter C, Part 742, §742.15.
[40] "itunes connect developer help: Provide export compliance documentation." https://help.apple.com/itunes-connect/developer/#/dev88f5c7bf9. Accessed 2017-12-28.
[41] "Play console help: Export compliance." https://support.google.com/googleplay/android-developer/answer/113770. Accessed 2017-12-28.
[42] For reference, the most powerful DoS attack ever recorded struck www.github.com with the power of 1.35 terabits per second of traffic (many times more powerful than a 2015 Chinese state sponsored hack), and was completely thwarted in under half an hour: https://www.wired.com/story/github-ddos-memcached/

signed. While Android does not offer such built in protection, similar approaches can be taken to address tampering issues: verifying the application's current code signature manually at runtime, verifying that the application was delivered from the Google Play store (which should be the only acceptable installation vector), and ensuring the application is not running in an emulator.[43] Preventing runtime binary modification falls within our purview as well. Both iOS and Android client applications must detect the presence of a debugger if the application is not being run in a permissible build configuration.

Potential modification, tampering, or debugging of a live application should result in the creation of an audit entry that may be reported back to the web service. Once that request has been sent, the application should prevent any further action from being taken in the application and alert the voter that a potential attack has been detected.

See Application Counterfeiting & Imposter Clients for further discussion.

Data Exposure: Protecting any temporary data at rest, as well as ensuring data is not persisted longer than necessary are vitally important to the privacy of the voter and a cast vote record. As such, any client application database which is maintained should feature full encryption. SQLite does not offer full database encryption as a feature out of the box —- therefore, extensions to SQLite (such as SQLCipher) should be used to encrypt a database using a key that could be provided by the voter. That database key must never be hard-coded in the application. The voter's saved credentials must never be stored using methods that could potentially lead to that information becoming accessible to other applications on the system. This voter data, in addition to database encryption keys, and any additional data which is best suited for storage outside of a relational database should be persisted using secure data store systems such as the Keychain system in iOS or the Keystore system in Android.

4.6. Identifying Malicious Actors and Activity

The Proof of Vote protocol has built-in protection from malicious nodes actively trying to disrupt the election and identifying related malicious activity.

Malicious node identification happens as part of transaction confirmation. Every single transaction that is verified goes through a two-step process on every consensus participant in the network:

Step 1: Verify if the transaction is valid.
For votes, this would be determined by:

- ensuring a voter's pseudonym is part of the *Initialization Transaction*
- that a *Revocation Transaction* has not cancelled the *Initialization Transaction*
- that no other *Vote Transaction* exists for this pseudonym. Note that some elections may allow for multiple *Vote Transactions* per voter, with only the last one being tallied. If this is the case, validation only requires that there be no other Vote Transaction for this pseudonym for the current block.
- that the Zero Knowledge Proof provided proves that the encrypted Submission Style ID matches the provided Submission Style ID on both the *Vote Transaction* and the relevant *Initialization Transaction*.
- that the signature on the *Vote Transaction* is the valid and matches the public key found in the relevant *Initialization Transaction*

[43] S. Alexander-Bown, "Android security: Adding tampering detection to your app." https://www.airpair.com/android/posts/adding-tampering-detection-to- your-android-app.

Step 2: If the transaction is invalid, determine if:

(a) The transaction is invalid because the verifying node does not have enough information to verify the transaction, OR
(b) The verifying node has all information required to verify the transaction, and the transaction is demonstrably invalid.

When a verifying node notes a transaction that is demonstrably invalid, it takes note of which node proposed the transaction and marks the node as potentially malicious.

External identification of malicious nodes. Malicious nodes may also be identified from outside the Protocol. As an example, the election authority may notice that a node is consistently reporting to voters that their vote has been accepted, but not broadcasting the vote transaction to other nodes for inclusion on a block. This can be identified by a voting client's reporting of non-existent blockchain confirmations after vote submission acceptance.

4.7. Quantum Computers

Quantum computers represent a serious security concern for all digital systems, including those used for elections. Estimates for when a quantum computer capable of compromising modern cryptography might be built varies wildly. While most estimates see it taking a couple decades, Mariantoni suggests it might be possible for a nation state to build a quantum computer capable of breaking RSA-2048 within 15 years due to advances in quantum error correction.[44] The effect of quantum computers on modern cryptography can be broadly broken down into three groups:

1. Algorithms that can be completely broken by a quantum computer via Shor's algorithm and include RSA, ECC, ElGamal, and Pallier.
2. Algorithms that are partially compromised by a quantum computer via Grover's method and include AES, SHA, and HMAC. These algorithms can be safely used in a post-quantum world by doubling their key length.
3. Algorithms that are unaffected by any quantum algorithm and include Polys1306, GMAC, and hash-based signature schemes.

To guarantee the soundness of cryptographic systems in the quantum era, any proof of validity for operations like proof of shuffling in our e-voting system must be guaranteed without using cryptographic primitives that are amenable to Shor's algorithm. If using a cryptographic primitive that is amenable to Grover's method, the key length should be doubled.

While most symmetric ciphers are robust against Shor's algorithm, exchange/distribution of their keys still depends on asymmetric-encryption-based secure communication channels or protocols like Diffie-Hellman key agreement, which depends on the same computational assumptions as the asymmetric encryption algorithms. Quantum Key Distribution (QKD) claims to be guaranteed by the laws of physics and thus solves the problem physically. However, the hardware and networking requirements for practical QKD are still an unsolved challenge. Instead, we will explore classical post-quantum public key encryption algorithms for distributing of symmetric keys.

[44] M. Mariantoni, "Building a superconducting quantum computer." https://www.youtube.com/watch?v=wWHAs–HA1c, 2014.

Quantum Computers Impact on Proof of Vote®

Impact on the cryptography employed in Proof of Vote including hash functions, encryption algorithms, digital signatures, key generation mechanisms, ZK proofs and mix networks are discussed respectively in this section.

Impact on Hash Functions. SHA256, the hash function used throughout the protocol, provides 256 bits of classical preimage security and 128 bits of classical collision security. Using Grover's method, quantum preimage security is reduced to 128 bits.[45] For collisions, quantum computers do not do as well; a classical birthday attack takes $O(\sqrt{N})$, and a quantum birthday attack via Brassard–Høyer–Tapp[46] (based on Grover's method) takes $O(N1/3)$, providing 86 bits of collision security for SHA256. To achieve at least 128 bits of both preimage and collision security in a post-quantum world, SHA512 can be used in place of SHA256.

Impact on Encryption Algorithms. Proof of Vote makes use of ElGamal asymmetric encryption for vote privacy. ElGamal can be entirely broken by a quantum computer using Shor's algorithm. A post-quantum Proof of Vote protocol will need to transition to a post-quantum cryptosystem for vote privacy.

Impact on Distributed Key Generation. Distributed key generation for threshold encryption suffers from quantum attacks as the secret share-generating polynomial must be committed in some way in the existing solutions to provide verifiability. As any commitment is only computational regarding either privacy or soundness, so is the key sharing mechanism based on it. Although a novel verification mechanism free of any commitment can be designed to achieve both security properties unconditionally, there is another problem — the secret key shares must be distributed among the key holders. As any secure distribution channel depends on encryption in practice, the threat of quantum computers still cannot be completely removed until post-quantum public-key encryption algorithms become available.

Impact on ZK Proofs. Classic ZK proof is unconditionally sound with an overwhelmingly large probability and so its soundness (guarantee of the proved knowledge) is not affected by quantum computation. When we call a ZK proof protocol honest-verifier ZK or information-theoretic ZK, we do not mean that their privacy is guaranteed against quantum computation attacks. What we mean is that the proof protocol does not provide more chances to any attack against privacy than what already exists against the commitment of the secret knowledge (Any secret must be committed to in some kind of commitment so that ZK proof of it can be given on the basis of the commitment). If privacy of the commitment has already been compromised by quantum computers, it is senseless to continue with the ZK proof. Interactive ZK proof protocols are widely known to be limited in applications as they are not transferable and cannot benefit all potential verifiers. The cryptographic techniques to make a ZK proof protocol non-interactive are also affected by quantum computers and needs our consideration.

Impact on Mix Networks. Mix networks have the two security properties of soundness (to guarantee that the messages in the output ciphertexts is a permutation of the messages in the input ciphertexts) and privacy (confidentiality of the permutation and the shuffled messages). As discussed above, privacy depends on the employed encryption algorithm and so is vulnerable to quantum attacks. The concern for privacy is especially serious in re-encryption mix networks as all the existing re-encryption mix networks employ the classic public key encryption algorithms vulnerable to quantum attacks. If post-quantum asymmetric ciphers (ex: pre lattice-based ciphers) cannot be employed, decryption mix network based on symmetric encryption chain instead of the more common re-encryption mechanism must be adopted. However, decryption mix network is not verifiable unless the

[45] Amy, Matthew, et al. "Estimating the cost of generic quantum pre-image attacks on SHA-2 and SHA-3." *International Conference on Selected Areas in Cryptography*. Springer, Cham, 2016.
[46] Brassard, Gilles, Peter Hoyer, and Alain Tapp. "Quantum algorithm for the collision problem." *arXiv preprint quant-ph/9705002* (1997).

messages and permutation are revealed, so it cannot provide end-to-end verification with the overwhelmingly large probability desired in e-voting systems. In theory, the soundness of mix network is not affected by quantum attacks as it is unconditional and only bound by a probability, although as mentioned before decryption mix network cannot provide soundness efficiently by an overwhelmingly large probability. However, this assumption does not consider efficiency, another desired property of mix networks. In reality, most mix network designs needs some computational hardness for soundness. The challenge is to design an efficient mix network that does not depend on a quantum-vulnerable algorithm for it's soundness, so that at least the validity of tallying, and thus result of election, are not affected by quantum computers even if privacy of the votes may be breakable.

Post-Quantum Proof of Vote®

While not an imminent concern, quantum computers represent a considerable future threat. We propose a stepwise solution to the threat of quantum computers, gradually hardening future versions of the Proof of Vote protocol against quantum computers with a focus on prioritizing soundness over anonymity.

Step 1: Layer a post-quantum signature scheme (for example SPHINCS-256) on top of an existing RSA signature scheme (Figure 12). In this scheme, the public and private keys are compound keys, comprised of both the post-quantum SPHINCS-256 keys and the classically secure RSA keys. Publishing a public key would involve publishing both keys together as a package. Signing is done by signing the message twice, once with each key — a signature is therefore a compound SPHINCS-256/RSA signature. Verification is done by verifying each signature component against the corresponding public key component, with both signature components needing to be valid for the signature to be considered valid.

Step 2: Move to a post-quantum encryption scheme for privacy. When NIST publishes its recommended post-quantum public-key cryptography standard[47] (which we will call NISTPQ for the purposes of this paper), we can replace our use of ElGamal and RSA if a quantum computer seems impendent. When making the decision to transition to NISTPQ, it will be important to consider the possibility that the new NISTPQ will not be as secure against traditional cryptanalysis as the tried-and-true ElGamal and RSA. Unlike signature schemes, we cannot combine NISTPQ with ElGamal because of the need to maintain a usable re-encryption mixnet via homomorphic encryption.

Step 3: Remove use of RSA signatures. Once post-quantum signature schemes are considered well-studied and their cryptanalysis is well understood, we can remove our legacy use of RSA signatures and move to an entirely post-quantum signature scheme.

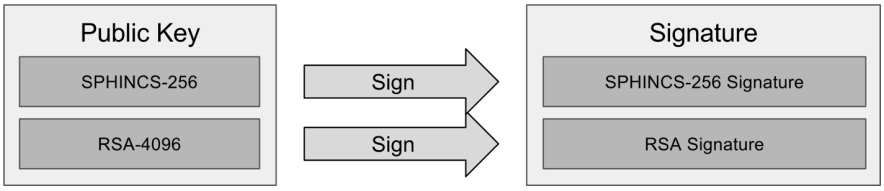

Figure. 12. *A interim hybrid signature scheme that combines the post-quantum SPHINCS-256 with the well-understood classical RSA.*

[47] See https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization

Proof of Vote®

A deeper discussion on:

- Cryptographic and Computational Security of
    - Distributed Key Generation
    - Re-encryption
    - Zero Knowledge Proofs
    - Interactive Device Challenges
    - Public Key Infrastructure
- Mix-Networks Implementation
- Details of Zero Knowledge Protocols Implemented
- Relation Attacks and Countermeasures
- Bypass Attacks
- Adjustments of Parameters of ElGamal Encryption
- Instantiated Batch Proof and Verification
- Computational Cost
- Other Concerns and Considerations

Can be found in the **Appendix.**

Please reach out to protocol@votem.com with any questions, comments, feedback, or concerns.