2023-03-25

PVPBC: Privacy- and Verifiability-Preserving E-Voting Based on Permissioned Blockchain

Muntadher Sallal
Bournemouth University, msallal@bournemouth.ac.uk

Ruairí de Fréin
Technological University Dublin, ruairi.defrein@tudublin.ie

Ali Malik
Technological University Dublin, ali.malik@tudublin.ie

Recommended Citation

Sallal, M.; de Fréin, R.; Malik, A. PVPBC: Privacy- and Verifiability-Preserving E-Voting Based on Permissioned Blockchain. Future Internet 2023, 15, 121. DOI: 10.3390/fi15040121

Funder: Science Foundation Ireland

PVPBC: Privacy- and Verifiability-Preserving E-Voting Based on Permissioned Blockchain

Muntadher Sallal and Ruairí de Fréin and Ali Malik Ali
Ollscoil Teicneolaíochta Baile Átha Cliath,
Campas na Cathrach,
Ireland.
web: https://eleceng.tudublin.ie/rdefrein/

BibTeX:

```
@article{deFrein23PVPBC,
  author = {Sallal, Muntadher and de Fr\'{e}in, Ruair\'{i} and Malik, Ali},
  title = {PVPBC: Privacy- and Verifiability-Preserving E-Voting Based on Permissioned Blockchain},
  journal = {Future Internet},
  volume = {15},
  year = {2023},
  number = {4},
  article-number = {121},
  url = {https://www.mdpi.com/1999-5903/15/4/121},
  issn = {1999-5903},
  doi = {10.3390/fi15040121}
}
```

future internet

MDPI

Article

PVPBC: Privacy- and Verifiability-Preserving E-Voting Based on Permissioned Blockchain

Muntadher Sallal [1], Ruairí de Fréin [2,*] and Ali Malik [2]

[1] Department of Computing and Informatics, Bournemouth University, Dorset BH12 5BB, UK
[2] School of Electrical and Electronic Engineering, Technological University Dublin, D07 EWV4 Dublin, Ireland
[*] Correspondence: ruairi.defrein@tudublin.ie

Abstract: Privacy and verifiability are crucial security requirements in e-voting systems and combining them is considered to be a challenge given that they seem to be contradictory. On one hand, privacy means that cast votes cannot be traced to the corresponding voters. On the other hand, linkability of voters and their votes is a requirement of verifiability which has the consequence that a voter is able to check their vote in the election result. These two contradictory features can be addressed by adopting privacy-preserving cryptographic primitives, which at the same time as achieving privacy, achieve verifiability. Many end-to-end schemes that support verifiability and privacy have the need for some voter action. This makes ballot casting more complex for voters. We propose the PVPBC voting system, which is an e-voting system that preserves privacy and verifiability without affecting voter usability. The PVPBC voting system uses an effective and distributed method of authorization, which is based on revocable anonymity, by making use of a permissioned distributed ledger and smart contract. In addition, the underlying PVPBC voting system satisfies election verifiability using the Selene voting scheme. The Selene protocol is a verifiable e-voting protocol. It publishes votes in plaintext accompanied by tracking numbers. This enables voters to confirm that their votes have been captured correctly by the system. Numerical experiments support the claim that PVPBC scales well as a function of the number of voters and candidates. In particular, PVPBC's authorization time increases linearly as a function of the population size. The average latency associated with accessing the system also increases linearly with the voter population size. The latency incurred when a valid authentication transaction is created and sent on the DLT network is 6.275 ms. Empirical results suggest that the cost in GBP for casting and storing an encrypted ballot alongside a tracker commitment is a linear function of the number of candidates, which is an attractive aspect of PVPBC.

Keywords: verifiable voting; online voting; Selene; distributed ledger technology

check for updates

Citation: Sallal, M.; de Fréin, R.; Malik, A. PVPBC: Privacy- and Verifiability-Preserving E-Voting Based on Permissioned Blockchain. *Future Internet* **2023**, *15*, 121. https://doi.org/10.3390/fi15040121

Academic Editors: Christoph Stach and Clémentine Gritti

Received: 19 January 2023
Revised: 10 March 2023
Accepted: 21 March 2023
Published: 25 March 2023

1. Introduction

Many conventional offline services, such as voting, mail, and payments, are migrating online due to the rapid development of the Internet and information technologies [1]. E-voting is an area of research which is attracting significant attention. The uptake of e-voting has gradually spread throughout European countries –and to non-European countries– with successful outcomes. However, e-voting comes with its own security challenges which need to be overcome to safeguard the pillars of free-and-fair election processes. These challenges include ensuring: (1) strong voter authentication; (2) voter privacy; (3) end-to-end (E2E) verifiability; and, finally, (4) election transparency and integrity [2].

To design and implement a successful e-voting system, several requirements must be met. These requirements are named and described as follows:

1. **Eligibility:** each voter should only be allowed to vote once. Only eligible voters are allowed to vote;
2. **Fairness:** no early results can be declared during the election period which would influence others who have not voted yet;

3. **Individual verifiability:** the ability to check that the vote was correctly counted should be available to individual voters;
4. **Universal verifiability:** every independent person should be able to verify the operations performed at every stage of the election as well as to check that the published results actually sum to the total number of cast votes;
5. **Vote privacy:** how a voter voted should not be disclosed to anyone;
6. **Receipt freeness:** the system should not offer any information which voters can use to prove how they voted. This information is typically expressed in the form of a receipt;
7. **Coercion resistance:** the system should not allow the voter to present evidence to a coercer of how they voted.

We address the following problem. How can blockchain be used in an e-voting framework in a way in which voter usability is preserved, privacy is enforced, and, finally, verifiability is ensured? We hypothesize that a solution addressing this problem statement should consist of (1) a conceptual framework which is based on blockchain; (2) the use of novel cryptographic primitives and mechanisms as part of this framework to enable checks of individual and universal verifiability. Consequently, voters must be able to verify the availability and accuracy of their votes in the final tally and that the published results are correct (sum of all votes); and, finally, (3) safeguarding the voter's privacy during the authentication phase and post-election phase, as well as ballot privacy during the election phase. In our previous work, we put forth the VMV (Verify-My-Vote) voting system [3], which uses the Selene voting scheme as the underlying voting system to preserve verifiability. In this paper, we extend VMV by incorporating privacy and verifiability functionalities into existing e-voting platforms without affecting the voter's usability.

1.1. Motivation

Privacy is a crucial requirement in e-voting systems which ensures that the connection between voters and their vote remains hidden. Many e-voting systems achieve privacy by making use of several cryptographic mechanisms. The aim is to remove the linkability feature between a vote and the caster of the vote during all election phases. Preserving privacy during the voter-identification (ID) phase is a challenging task to achieve. This is because additional cryptographic arrangements must be established to prevent linking a voter's identity to the contents of their vote. Blind signatures and mixed-nets are among the techniques that can be applied to remove the connection between the voter and their ballot [4]. These arrangements may clash with voter-usability requirements, which can negatively affect voter turn-out.

Meeting the verifiable-e-voting requirement is important in reducing the need to trust electronic systems. Systems meeting this requirement afford voters and observers the ability to independently check whether the votes were recorded, counted, and tallied correctly. However, Internet-based voting systems in use today are not sufficiently robust to satisfy the verifiable-election requirement as they do not provide proof, by means of corroborating evidence, that would allow clear, individual and universal determination of the election's verifiability [5]. To ensure verifiability, many E2E methods necessitate voter involvement, such as executing the cut-and-choose action during the vote-casting phase. Cut-and-choose protocols require complex interaction with the voter (e.g., interactive proofs) to reduce the probability of corruption [6]. However, cut-and-choose protocols make the voting process more complicated for voters, negatively affecting their ease of use. Replacing existing systems with a brand new E2E verifiability scheme that supports voters usability presents a risk to businesses.

1.2. Contributions

To conceal the link between the voter and ballot during the authentication phase in such a way that a cryptographic action by voters is not required, an authentication protocol which keeps the voter's identity confidential is introduced in this paper. The protocol uses an access model which is based on the revocable-anonymity concept where incorporating identity revocation in an anonymous communication system is achieved using a permissioned distributed ledger (DL) and smart contract. To ensure resilience, trust, and privacy, e-voting requires a framework for access control which matches the decentralized attribute of e-voting.

To provide E2E verifiability, the Selene e-voting scheme [7] is employed as an e-voting scheme which is designed to support individual and universal verifiability. This means voters and observers can check the validity of votes during the election and post-election phases. Selene simplifies the voter's experience during the vote-casting time by hiding the cryptographic complexity from the voter. The proposed system uses DL technology within Selene to manage cryptographic primitives.

We summarise our contributions as follows:

- We propose an e-voting framework, which is called the PVPBC voting system, which preserves crucial e-voting features: (i) voter privacy and anonymity by making use of a permissioned ledger technology and smart contracts; and (ii) E2E verifiability by making use of the Selene voting scheme and a permissioned DL. Importantly, these crucial features are provided without affecting user experience, as the PVPBC framework does not affect the protocol fulfilled by the voter when they are voting.
- We propose a comprehensive architecture for a capability-based authorization protocol, which can be used to authenticate voters in e-voting platforms. The proposed approach supports dynamic voter authentication in e-voting systems based on an access-control model which supports revocable anonymity, as a result of using permissioned DLs. The protocol includes capability management and access-right validation.

Compared to existing work, the PVPBC voting system we introduce has a number of benefits, which we emphasize here: (i) it supports usability preservation for voters in the sense that voting is the only task required of the voter; (ii) the system integrity and security is supported by a distributed authentication mechanism based on DL technology; (iii) the system is verifiable by all parties; and (iv) the system fulfils vote privacy as well as voter privacy.

1.3. Organisation

The structure of this paper is outlined as follows. We start by reviewing the state-of-the-art in Section 2 and by identifying gaps in the current state-of-technology. In Section 3, we introduce the PVPBC voting system on a system-component level. We outline the assumptions made and discuss how authentication with revocable anonymity is achieved. Section 4 discusses the role of the permissioned distributed-ledger technology (DLT) in the context of the contributed e-voting scheme in terms of the front-end system and authentication. The voting experience/protocol followed by the voter at the pre-election phase, voting phase and also the tracking-number retrieval process is outlined in Section 5. Section 6 describes the technical details involved in the pre-election, election-phase and post-election-phase setup. A thorough analysis, under the headings: eligibility, privacy, integrity, fairness and verifiability, is provided in Section 7. This analysis is supplemented by a performance analysis in Section 8, which evaluates the registration and authentication schemes along with an evaluation of the voting-phase performance. We provide our conclusions and make recommendations for future research in Section 9.

2. Related Work

E-voting is attracting increasing research attention. The aim of this research has been to preserve the key attributes of e-voting which are privacy, integrity, transparency, and E2E verifiability. From the protocol point of view, preserving vote privacy is challenging when e-voting systems are being designed. Several mechanisms have been adopted in the literature to guarantee privacy in e-voting systems. One mechanism is to use blind signature schemes [8]. The distinguishing feature of these schemes is that the voter receives a token which includes a blind signature from the administrator as an indication of the voter's eligibility to vote. In response to this, the voter should send their vote anonymously along with the signed token as a proof of eligibility. An alternative approach is to use homomorphic encryption [9]. In this approach, the voter encrypts their vote. After that, the administrator determines the encrypted tally from the encrypted votes by making use of the encryption algorithm's homomorphic properties. Randomness, which is achieved using mix-nets, is also an exploited mechanism to preserve privacy by mixing up votes so that the connection between voters and their ballot is obscured [10] .

Homomorphic encryption is the foundational principle for a number of e-voting systems. In the system proposed in [11], the voter can encrypt their vote and generate a proof of validity which does not require interaction—this is referred to as a zero-knowledge proof– by making use of an El-Gamal scheme and a set of private and public keys that are shared between parties. A coalition of honest authorities can check the proofs from the voters and then combine all correct encrypted votes in order to decrypt them utilising the proofs. Even though most of the security requirements are fulfilled, the complexity of the proofs as well as the form of the votes affects the scalability of the system.

Several proposals for E2E-verifiable voting systems, which employ common verifiability techniques to support the integrity, and thus the credibility of the election, have been introduced. Selected examples include Prét à Voter [12], Wombat [13], Scantegrity II [14], Helios [11], Belenios [15], Civitas [16], and the Selene protocol [7]. These techniques are suitable for either paper voting or electronic voting. Recent verifiability techniques target remote voting from the voter's device.

The Benaloh Challenge, which is discussed in [17], is a typical mechanism to confirm that a vote is cast according to the voter's intent where a cut-and-choose method is applied to ensure that the vote is accurately constructed. After creating the vote and encrypting it, the voter has two options, either to cast the vote or to audit it. The auditing process involves the disclosure of the vote, without casting it, and offers proof that the vote was accurately constructed. The voter is able to carry out vote auditing several times before submitting an un-audited vote. This action is achieved by using one of the following algorithms, the Helios, Belenios, Civitas or the Wombat algorithm.

Pretty Good Democracy [18] is a verifiable e-voting system which makes use of a Code-Vote approach as a way to obtain individual verifiability. A code sheet is given to the voter in this approach, which contains a voting code for each candidate, as well as a return code. These codes are delivered to the voter using a private channel such as the post. During the voting phase, the voter casts their vote by submitting the code of their candidates. Using the return code, the voter extracts verification that the vote has been correctly received. This is possible because only the election system has knowledge of the voting codes and the return codes.

In recent years, several Internet voting systems have been proposed that either use DL technology or have been launched as DL-based systems [19–22]. Some leading examples of live systems include, Follow My Vote (https://followmyvote.com/, accessed on 18 January 2023) and Democracy.Earth (https://www.democracy.ea, accessed on 18 January 2023). Typically, these proposals view the act of casting a vote as a transaction which should be documented on a blockchain or DL. A systematic review of the literature on blockchain-based solutions for e-voting and an analysis of the techniques reviewed is given in [23]. Blockchain is reviewed from a security perspective in [24]. The authors perform their analysis at three levels: the process, data and infrastructure levels and emphasize the need to consider these levels in light of urgent business and industrial concerns. Occasionally, the act of casting a vote is analogous to transferring a vote "coin" to a specific candidate. Nevertheless, these proposals often fail to tackle concerns related to electronic voting. These concerns include, but are not limited to, ballot secrecy, E2E verifiability, resistance to coercion, confirmation of accurate casting and recording of votes, voter eligibility, and the guarantee that all ballots cast before the end of the election will be counted. In addition, these actions could have unfavorable consequences, such as disclosing real-time running totals or voter turnout during the election, or linking a monetary value to a vote. Limited levels of technical details are available for some of these systems. The technical details of the schemes that are available are difficult to review and to independently assess. On the other hand, the VMV voting system introduced in [3] also uses the Selene voting scheme as the underlying voting system and employs blockchain to preserve verifiability. However, VMV sacrifices the voter-privacy property as the EA has full control of the voter's identity. VMV was added as a verifiability layer on the already-available legacy system.

A solution for the integrity and verifiability problems is commonly sought by applying the DL concept. Adoption of a permissionless DL on public, peer-to-peer networks gives rise to a number of security and performance issues which have not been addressed in these solutions. Firstly, inconsistency in the DL, which might result as the necessary and sufficient conditions for independent consistency verification, is not well-defined or understood. Secondly, due consideration is not given to information propagation delay, which is an on-going concern in public networks that maintain DLs [25,26]. Potential solutions exist in the form of modern machine-learning approaches that aim to minimize variation in propagation times [27,28] and deep-learning-based traffic-classification schemes [29] which seek to utilize network resources better. The net result of these issues is that votes might be absent from the DL at the end of an election. Finally, performance issues, which are related to transaction costs, are high when a public DL is used.

In summary, we compare the shortcomings and strengths of state-of-the-art e-voting systems in Table 1 under the headings Eligibility (El), Privacy (Pr), Verifiability (Vr) and Usability (Us), in order to establish the need for PVPBC.

Table 1. Motivating the need for PVPBC by comparing the strengths and shortcomings of existing e-voting systems under the headings: Eligibility (El), Privacy (Pr), Verifiability (Vr) and Usability (Us).

Voting Scheme	El	Pr	Vr	Us	Limitations	Strengths
Prét à Voter [12]	✓	✓	✓	×	Secrecy of the election is at risk when the election authority is compromised. Uses cut-and-choose protocols which require complex interaction with the voter.	Offers verifiable elections.
Wombat [13]	×	✓	✓	×	Paper-and-cryptographic-based voting system. Usability issues identified during the trials which arise due to the ballot design.	Overcomes the privacy issues associated with on-demand print ballots.
Scantegrity II [14]	×	✓	✓	×	Does not support verifiability in remote voting. Uses cut-and-choose protocols which require complex interaction with the voter.	Individual verifiability is achieved by a cut-and-choose mechanism.
Helios [11]	×	✓	×	×	Does not support verifiability in remote voting. Requires a separated public authentication-mechanism service. It is vulnerable to ballot stuffing.	Offers verifiable online elections.
Pretty Good Democracy [18]	✓	×	✓	×	Privacy is at risk as the election system has knowledge of the voting codes associated with the verifiability process.	Individual verifiability is achieved by making use of a code-vote approach.
Belenios [15]	✓	✓	✓	×	Not suitable for use in high-stake elections as it is not coercion-resistant.	Offers verifiable online elections.

Table 1. *Cont.*

Voting Scheme	El	Pr	Vr	Us	Limitations	Strengths
VMV [3]	✓	×	✓	✓	The voter-privacy property is at risk as the EA has full control of the voter's identity.	Offers verifiable and usable online elections.
DLT systems [19–22]	✓	×	×	×	Fails to tackle concerns related to electronic voting. These concerns include, but are not limited to, ballot secrecy, E2E verifiability, resistance to coercion, confirmation of accurate casting and recording of votes, voter eligibility, and the guarantee that all cast ballots before the end of the election will be counted.	Offers faster Internet voting based on blockchain.

3. Core of the PVPBC Voting System

The key innovation of the proposed Privacy- and Verifiability-Preserving E-voting Based on Permissioned Blockchain (PVPBC) voting system is to introduce a fully verifiable e-voting protocol. The PVPBC voting system should provide the verifiability, voter usability, and integrity properties. The PVPBC voting system is comprised of three components which are: the front-end system; the election authority (EA); and finally, the trusted third party (TTP). It is a requirement that these components are independent of each other. To satisfy the aim of implementing E2E verifiability, the PVPBC voting system uses the Selene e-voting scheme [7]. As a consequence of using the Selene protocol, the front-end system has two distinct, desirable characteristics. Firstly, the voters' cast ballots are gathered and presented on a permissioned distributed ledger in an unencrypted format, and then matched with anonymous tracking numbers. This approach is distinct from many other e-voting systems described in the literature, which only disclose an encrypted or hashed rendition of the votes on the bulletin board (BB), (an area within the e-voting system which includes the final election results). By publishing plaintext votes on the DL, voters can be confident that their vote was accurately recorded and tallied. It follows that voters have no need to trust or possess any specific understanding of the cryptographic methods utilized in the election. Publishing votes in plaintext on the DL guarantees individual verifiability. In effect, any voter can check their intended vote against the plaintext vote recorded in the DL by using their tracking number. Universal verifiability is also guaranteed because the tally can be calculated by anyone using the plaintext votes. The PVPBC voting system also provides mechanisms to safeguard against coercion, as a result of using the Selene protocol. These mechanisms are available to voters if they experience coercion.

To achieve the objective of providing access-control processes which are effective for voting services and information in e-voting systems, the PVPBC voting system offers an anonymous authentication mechanism where the voter's vote can be revoked by the election service provider with the cooperation of a trusted organization. An identity-based capability-token management strategy which is robust is proposed, based on the blockchain network. The use of smart contracts is instrumental in achieving effective registration, propagation and the revocation of access authorization. Consequently, election service providers are able to verify eligible voters without knowing their real identity. Voters only reveal their identity to the trusted organization as a TTP. During the election period, the TTP collaborates with the election service providers to validate voter eligibility without revealing their identity to the election services. This authentication feature is implemented in the PVPBC voting system by making use of a permissioned DL and smart contract. The permission DL is implemented by several trusted authorities within the trust organization, to store and initiate capability access tokens for eligible voters.

3.1. System Component Overview

Figure 1 illustrates the PVPBC voting system's components. Further explanation of how these components work will be provided in Section 4, but a brief overview is provided now in order to provide a working knowledge of the system.

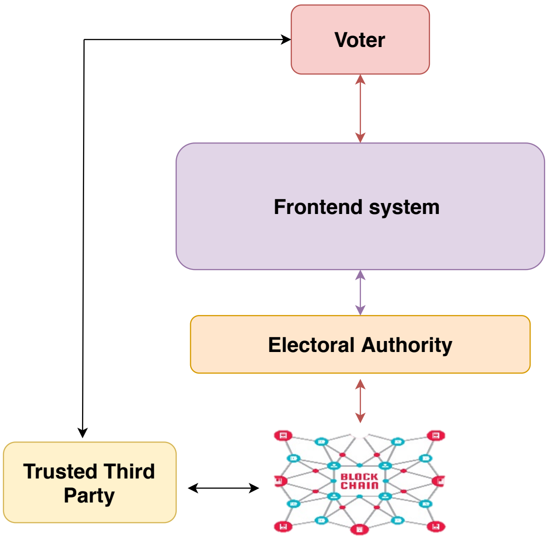

Figure 1. Component overview for the PVPBC voting system and interactions between the election actors: the election authority, the trusted third party, the voter and the front-end system. PVPBC is supported by blockchain.

Trusted third party (TTP): an individual or organisation which ensures that the election is correctly held by making use of DL technologies and smart contract to verify eligible voters.
Front-end system: a self-contained component which provides a verifiability functionality similar to the Selene protocol. A crucial difference from the Selene protocol is that the cryptographic functionality is provided by the front-end system, which includes key generation, signing, and decryption for voters. As a result, these functionalities do not have to be provided by the voters. This is managed independently of the EA. The front-end system also provides the authentication and voting services for voters. Specifically, the front-end system verifies voter-authentication requests and allows eligible voters to access the ballot and to cast their votes.

The front-end system is composed of several components. These components collectively provide secure and verifiable voting.

- **Voter-key management:** This component manages voter keys and their application. This is run, not by the EA, but by an independent trusted party.
- **Tellers:** generation and management of the election threshold key and the cryptographic manipulations required by the Selene protocol is performed by a set of tellers [30].
- **Web bulletin board (WBB):** To distribute trust, the front-end system utilizes a permissioned DL with a group of peers. This process requires a consensus of over two-thirds majority for agreement on the contents of the ledger.

Voters: Voters interact with the registration website offered by the TTP. Other interactions with the voters are typically via email as well as a voting website which is offered by the front-end system. In our proposed system, voters are not required to handle their own keys. Instead, the voter-key management component manages this for the voters, and voters use the credentials provided to access the required cryptographic functions. The voter-key management component is run by trusted parties within the front-end system.
Election authority: the EA is in charge of setting up the election, running it, and verifying voter eligibility in collaboration with the front-end system and the TTP.

Authentication-permissioned DLT: The authentication DLT manages voter authentication by making use of capability tokens, which are embedded in a smart contract. The authentication DLT is maintained by the TTP.

3.2. Assumptions

We make the following assumptions. We assume that:

1. There are authenticated channels from the front-end system to the authentication server of the EA.
2. There is a secure channel from the EA to the TTP.
3. Tellers are trusted parties who collaborate with the EA to run the election.

3.3. Authentication with Revocable Anonymity

The PVPBC voting system is based on a brand-new voter-authentication mechanism which preserves voter privacy and enforces the legal requirements of the election service authorities. To verify voters, the PVPBC voting system uses TTP certification along with DL technology and a smart contract. The TTP issues a capability access token for each eligible voter. The DL technology holds the access tokens securely and releases them when certain requirements have been fulfilled. At least five parties participate in the protocol, they are listed now. The TTP is comprised of two parties, the T_A, which is a trust body within the organization, and the T_m, which is an administrator. The third party is the voter, V. The permissioned DLT is run by two additional parties, an administrator, T_m, and several T_As.

Initialisation phase: Prior to the election, the T_m maintains a list of ID numbers, which belong to eligible voters. It is populated on the DLT by the TTP administrator T_m. Once the ID number related to the voter is verified, a profile for the registered voter is created on the DLT by calling the *Register Smart Contract* (more details are provided in Section 6.1). Each voter profile on the DLT includes the registration information tuples (Voter ID number (VID), Flag—"Not voted"). The VID is a unique virtual ID, which is assigned for each registered voter on the DLT.

Smart-contract deployment: A smart contract, which oversees the capability access token for each eligible voter, must be deployed on the DLT by the TTP. The smart contract can securely manage any algorithmically specified protocol, thanks to cryptographic protocols, when it is deployed over the DLT network. The smart contract supplies a deployAction () ABI (smart-contract interaction interface) for this function. The permission to execute this ABI rests with T_m. When operating this ABI, the T_m must send a transaction which includes the voter information defined in the previous phase, namely, the initialization phase. After synchronizing the permissioned DLT data, any node in the permissioned network, for example, the T_m or the T_A, has the ability to communicate with the smart contract using the contract address provided.

Authorization validation: The authorization-validation process takes place at the election service provider, in response to receiving an authentication request from the voter during the pre-election phase. More precisely, as shown in Figure 2, the election service provider verifies the present condition of the smart contract in the permissioned DLT to obtain the capability token associated with the voter's VID. The smart contract furnishes an accessRequest() ABI for verification of the token. The election service provider can accomplish this ABI by submitting the pertinent information, the voter's VID and the action to be performed using the transaction. The transaction is mined, included into a block and broadcast to the nodes, T_m and T_A, in the TTP DLT network. As part of this process, each node that receives the transaction carries out the ABI to check if the voter possesses the necessary access rights. This assures system users that nodes cannot deceive others with false processing results. The result is a robust and trustworthy access-control system. The election service provider will grant access to the ballot to the voter if the access-right policies and conditional constraints are satisfied and also depending on the capability-token validation and access-authorization-process result. If these conditions are not satisfied, the voting request is denied.

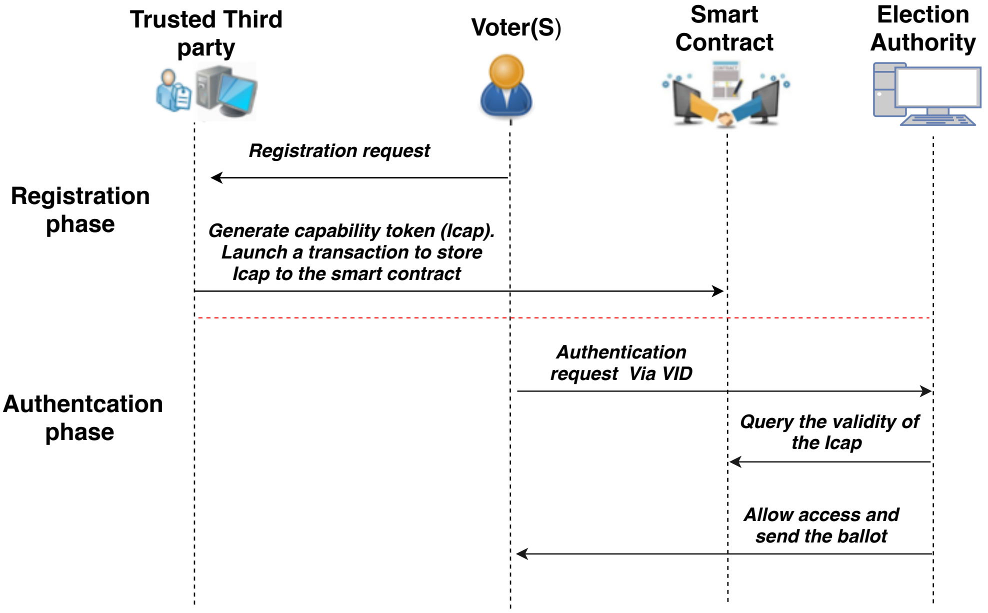
Trusted Third party
Voter(S)
Smart Contract
Election Authority
Registration phase
Registration request
Generate capability token (Icap). Launch a transaction to store Icap to the smart contract
Authentcation phase
Authentication request Via VID
Query the validity of the Icap
Allow access and send the ballot

Figure 2. Voter-access control is carried out using two phases: the registration phase and the authentication phase. Effective access-control processes for voting services and information in e-voting systems is achieved by PVPBC by offering an anonymous authentication mechanism where the voter's vote can be revoked by the election service provider with the cooperation of a trusted organization.

Capability Token Structure

The T_m is responsible for maintaining a profile record for eligible voters on the DLT. In the profile record, all registered voters are associated with a virtual identity (VID) which is globally unique (we use two forms of ID. The voter ID is the normal identification required at the registration phase. It is typically a name or an email address. VID is another form of voter ID on the DLT. VID is a key which identifies the eligible voter on the authentication blockchain.), which is used as the prime key for identifying voters'off-chain profile. It is maintained by the TTP. At least one main account belongs to each voter. It is indexed by its VID address in the permissioned DLT. Consequently, the DLT is used to capture the VID for profiling registered voters. Generally, each eligible voter should be assigned a capability access token, which is stored in a smart contract. The capability token specifies the access rights associated with the voting ballot. It also contains awareness information. The capability token uses the following parameters.

- F: a cryptographic hash function which operates in one direction only;
- VID_V: the virtual ID of an eligible voter which solicits access, with the purpose of taking part in the election;
- EXP_R: the token expiration time;
- AR: the set of access rights pertaining to a predetermined set of actions;
- T: a timestamp;
- n_i: the tracking number for registered voters. This tracking number is a sparse selection of integers.

The capability token is then defined as a function of these parameters,

$$Token_R = F(VID_V, EXP_R, AR, T, n_i). \quad (1)$$

In the adopted authentication mechanism, the capability token corresponding to a particular voter is identified by the VID of the voter. Example actions in the set of access rights, AR, are the *vote* and the $NULL$ actions. If the access-rights set is null, $AR = NULL$, the voting operation is not allowed.

4. Role of Permissioned DLTs

The PVPBC voting system establishes two separate permissioned DLTs, each of which is used at a certain stage of the election to fulfill certain features and functions within the proposed system.

4.1. Election Permissioned DLT in the Front-End System

The front-end system makes use of DL technology. This DL is integrated into the Selene protocol as a key element to obtain the necessary assurances ahead of the election, e.g., the first half of commitments to tracking numbers. It is also responsible for storing the election's verifiability evidence as it is generated, and it handles the mixing and decryption of the votes and tracker numbers at the end of the ballot. This is carried out in this manner to ensure that the commitments that were placed in the ledger before the election started do not change after the election. Using the ledger guarantees that the same verifiability information is seen by all observers, and that the verifiability information cannot be changed at a later point in time.

In the front-end system, a permissioned DL is used. It is considered to be the appropriate design choice. This is because the DL is purpose-built for the election. In their current forms, EAs are necessary to manage the election roll. The need for the EA motivates the requirement for having trusted authorities. The EA's role is to start the election and to take responsibility for the election outcome. A benefit of the proposed system is that only the individuals responsible for conducting the election, who are already trusted, need to be granted write access to the ledger. Consequently, malicious nodes do not have the ability to take part in the DL's network. As a result, the security of the proposed system is increased. An additional advantage of permissioned DLs is that they permit faster and lighter consensus than the permissionless DLs. One side effect of the permissionless public-ledgers approach is that they give less control over consistency. This is because they support *eventual consensus*, which means that forks are possible and are eventually resolved. A good example of this is the branch rule of Bitcoin [31,32].

4.2. Authentication Permissioned DLT

The proposed system makes use of a permissioned DLT to manage and to store the authentication information. The authentication DLT is different from the election DLT, which is used to manage the verifiability data within the front-end system. The authentication DLT is maintained by trusted parties within the trusted third party organisation. This means that only those authenticated and trusted parties can join the network and can securely add, delete, and manage all voter information. During the registration phase, a TTP will be responsible for adding the voter information to the DLT as well as for deploying the smart contract related to the capability access token. We use the DLT here to evaluate the access policies, to preserve the access token integrity and to detect token double spending.

5. Voting Experience

The voting experience has three distinct stages: the pre-election phase; the voting phase; and, finally, the tracking-number retrieval phase. Each stage is now described.

5.1. Pre-Election Phase

Our starting point is the assumption that the EA oversees voter registration. It maintains the list of eligible voters. The EA also provides precise instruction, extensive advertisement and distribution, and election assistance. In Figure 3, we illustrate that voters need to remotely register with the TTP before the election. The voter needs to provide an ID key, e.g., national ID, in addition to other information, such as their e-mail address, phone number, and address. In response, the voter is given a unique pseudonyms key (VID) which can be provided during the voting phase for authentication purposes.

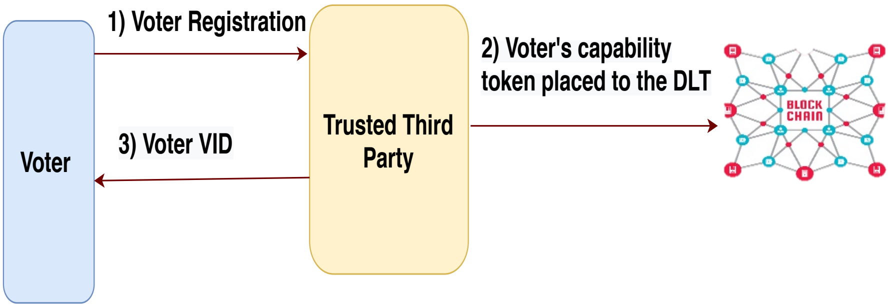

Figure 3. Voter experience during the registration phase: voters need to remotely register with the TTP prior to the election. The voter is given a unique pseudonyms key, which is used to provide authentication during voting.

During the pre-election phase, which is illustrated in Figure 4, voters need to authenticate themselves with the front-end system using their VID, which is generated during the registration step. Specifically, the voter, while interacting with the front-end system, is asked to input their VID, which is linked to the capability token stored in the authentication permissioned DLT. Upon retrieving a valid access token, the voter’s cryptographic primitives (voter’s signing keys, and voters’ trapdoor keys) are created and a tracker commitment is placed on the DLT. We describe the voter’s cryptographic primitives and tracker commitment in more detail in Section 6. Once the authentication stage has been successfully navigated, the voter is forwarded to the voting webpage.

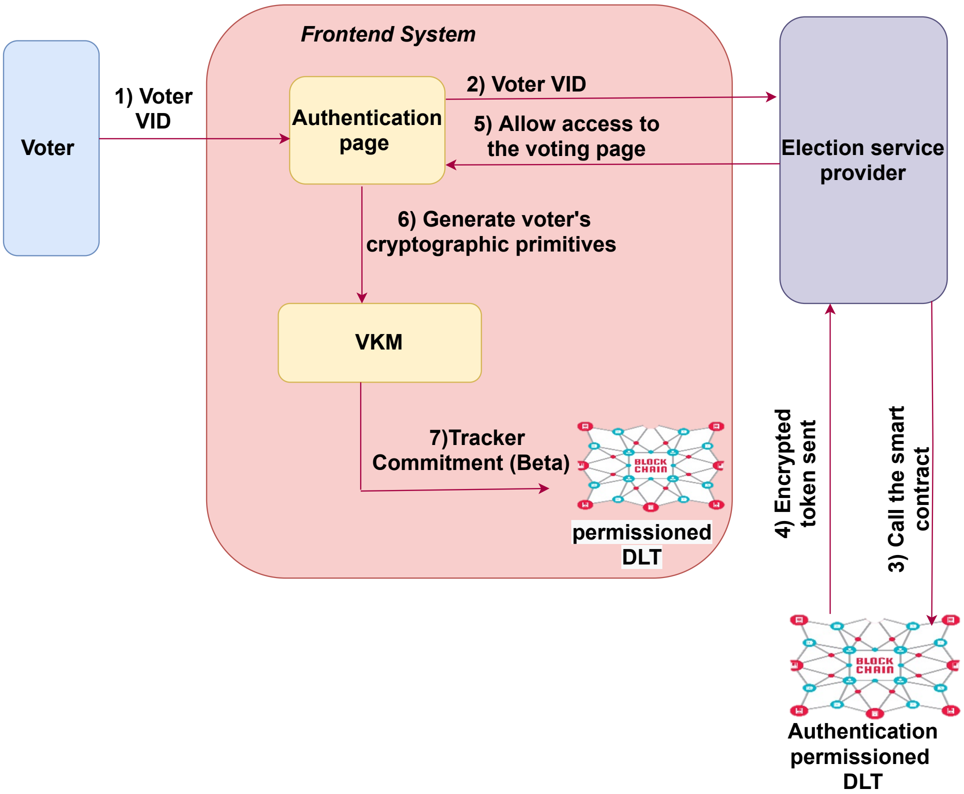
Frontend System
Voter
1) Voter VID
Authentication page
2) Voter VID
5) Allow access to the voting page
Election service provider
6) Generate voter's cryptographic primitives
VKM
7)Tracker Commitment (Beta)
BLOCK CHAIN
permissioned DLT
4) Encrypted token sent
3) Call the smart contract
Authentication permissioned DLT

Figure 4. Voter registration is facilitated by the interaction with the front-end system. Subsequent interactions with the election service provider, the authentication permissioned DLT and VKM are instigated by the authentication page in the front-end system.

5.2. Voting Phase

The casting of votes takes place during the voting period. After passing the authentication stage, which is illustrated in Figure 5, the voter is forwarded to the voting webpage. The voter is presented with choices and is prompted to make a selection(s). After the voter has confirmed their selection(s), they may proceed to submit their ballot. A ballot, in encrypted form, which contains the voter's choices, is sent to the authentication server within the EA for checking and signing. The server checks whether the ballot is accurately formatted. An example inspection which is carried out examines if the ballot has the accurate election identifier tag. When the ballot is correctly formed, the authentication server responds to the front-end system with an acknowledgement. This acknowledgement contains EA's signature on the ballot. Ultimately, the signed encrypted ballot is paired with the encrypted tracking number and stored on the DLT within the front-end system.

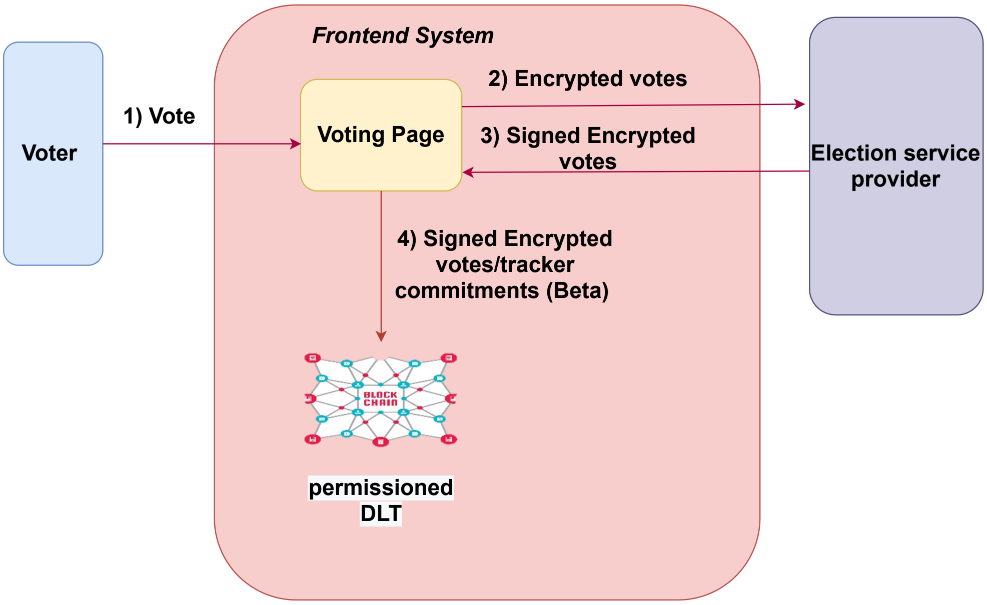
Frontend System
Voter
1) Vote
Voting Page
2) Encrypted votes
3) Signed Encrypted votes
Election service provider
4) Signed Encrypted votes/tracker commitments (Beta)
BLOCK CHAIN
permissioned DLT

Figure 5. Voter experience during the voting phase: Once authentication is complete, the voter is forwarded to the voting webpage. The voter is shown the choices and is allowed to make their selection(s).

5.3. Post-Election Phase and Tracking-Number Retrieval

Figure 6 illustrates the post-election phase processes. Once the election has ended, the front-end system WBB has the cast votes, in plaintext form, and the corresponding tracking numbers. Email or post is subsequently used by the TTP to send the α (trapdoor key which opens to a tracker β which is already posted on the DL) term to the voter. When the voter receives the α term, if they wish to verify their vote was cast as intended they can open their tracking-number commitment, access the information contained in the WBB and check their own vote in plaintext.

The voter's unique tracker is calculated by a supported web application within the front-end system using the received α term, the public β (a tracker commitment which is publicly assigned to each voter and paired with the encrypted and decrypted votes. α opens to a β, then the voter tracking number can be constructed) term, and the trapdoor key *sk*. The purpose of this action is to support individual verifiability.

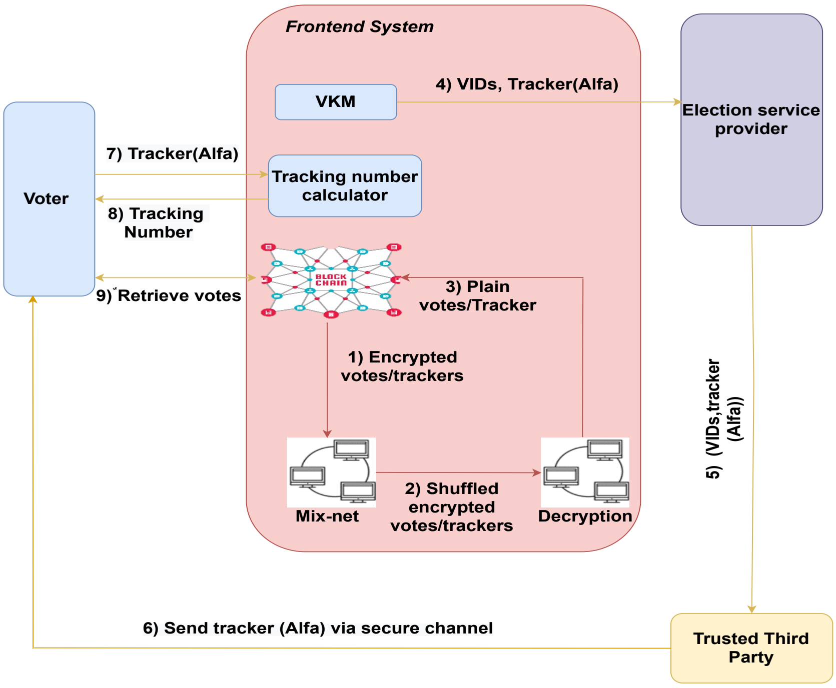

Figure 6. Voter experience during the post-election phase and tracking-number retrieval phase.

6. PVPBC Protocol: Cryptographic Considerations

We will now delve into a technical description of the protocol, paying particular attention to the cryptographic steps. This discussion is arranged into three parts: (1) the pre-election technical setup; (2) the election phase; and, finally, (3) the post-election phase.

6.1. Pre-Election Technical Setup

The pre-election setup phase consists of two main processes, a registration process and an election-key generation process. We describe voter authentication and the voter's cryptographic primitives in this context.

Registration: The voter-registration process is performed by the TTP by making use of a permissoned DLT and smart contract. Anyone with an ID number, for example, a national ID number, which is on the TTP's whitelist of IDs has permission to register for the election. When the ID field on the registration page is completed by the voter, as illustrated in Figure 7, the regApp.js application makes eth.calls to the registrar contract to check that the ID provided is on the whitelist of IDs, and if the associated voter has registered previously. When the outcome of the check is positive, the regApp.js communicates with the registrar contract to store the new voter information. This information consists of the voter's ID, Quorum DL address, and e-mail address. The purpose of this step is to link the user's Quorum DLT address and e-mail address so that they cannot register twice. Once registered, the registrar contract generates an access token for the voter and encrypts it with the private key of the TTP. The voter uses the access token for authentication on the blockchain. The voter is given a VID which is a parameter included in the access token generated by the smart contract (see Section 3.3 to obtain further information on this process).

Voter authentication: The voter interacts with the front-end system, providing their unique virtual identity (VID) to obtain entry to the voting platform, and then casts their ballot. Once voter authentication is complete, the voter should complete the unique virtual-identity (VID) field in the authentication page on the front-end system. After that, the authentication request and the voter's VID is forwarded to an election service provider. This process is illustrated in Figure 4. The election service provider first fetches the capability token from the smart contract by using the voter's VID, and then makes the decision whether or not to grant access to the ballot. The TTP changes the flag field of the voter's profile tuple to "voted" after successful authentication and updates the hashes of the blockchain. This prevents voters from voting more than once. Once voter access has been granted, the voter is directed to the voting page on the front-end system.

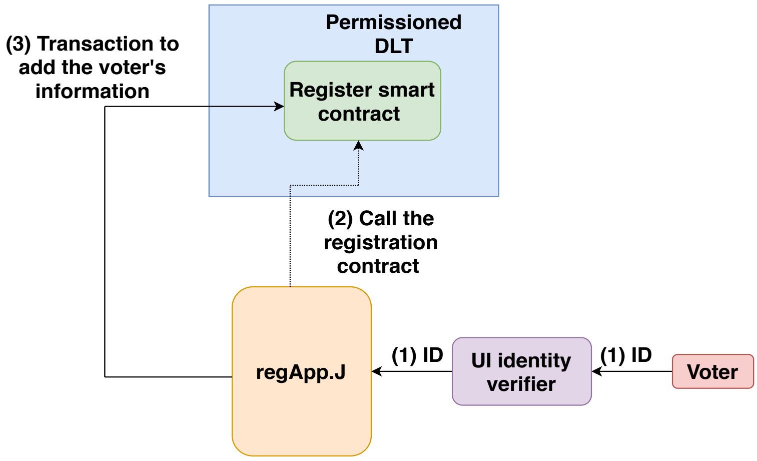

Figure 7. The voter-registration phase is managed by the TTP by making use of a permissoned DLT and smart contract.

Voter's cryptographic primitives: When the voter has been authenticated, their voter-keys management (VKM) component, which is part of the front-end system, generates cryptographic primitives for the corresponding authenticated voter. Cryptographic primitives include the voter's signing keys, and the voters' trapdoor keys. An ElGamal signing key pair, $(sign_i, vk_i)$, where $sign_i$ is the voter's signature key and vk_i is the public verifier key, is generated by the VKM component. The voter, V_i, generates signatures that can be verified via vk_i. It is assumed that a secret key, sk_i, and a public key (voter's trapdoor key), PK_i, can be generated by the VKM componentfor each voter, V_i, where $PK_i = g^{sk_i}$.

We move on to consider the creation of an encrypted tracking number for each authenticated voter. Voters obtain encrypted tracking numbers using the following procedure:

1. The tracking number, n_i, for the registered voter is extracted from the retrieved token.
2. The term, g^{n_i}, is calculated for the imported tracking number, n_i, in order to make sure that the tracking number falls in the appropriate subgroup.
3. Tracking numbers, g^{n_i}, are encrypted using the encryption key, PK_T.
4. The Sako–Kilian protocol [30] is used to re-encrypt and shuffle the set of encrypted tracking numbers, $(g^{n_i})_{pk_T}$. The resulting shuffled list is then assigned to the voters, so that each voter is associated with a unique secret encrypted tracker, $\{g^{n_{\pi(i)}}\}_{PK_T}$, where π is the permutation induced by the shuffle.

At this junction, the calculation of trapdoor commitments that open for a unique tracker occurs. The procedure by which the tracking-number commitment for each voter is generated is outlined.

1. A random number, r_i, is generated for each voter, i, by summing random values, $r_{i,j}$, which were generated by each teller, j, for voter i, where, $r_i = \Sigma^{i}_{j=1} r_{i,j}$.
2. The product, $\{PK_i{}^{r_i}\}_{PK_T}$, is generated using the component which is generated by each teller, j, $\{PK_i{}^{r_{i,j}}\}_{PK_T}$, and then by taking the product of these components. Similarly, $\alpha_i = g^{r_i}$ can be computed in a distributed fashion. The value, $\alpha_i = g^{r_i}$, will not be published, but it will be communicated privately to voters at the end of the election. In the PVPBC voting system, it is not computed until the end of the election.
3. The product of $\{PK_i{}^{r_i}\}_{PK_T}$ and $\{g^{n_{\pi(i)}}\}_{PK_T}$ is computed to obtain $\{PK_i{}^{r_i}.g^{n_{\pi(i)}}\}_{PK_T}$.
4. Threshold decryption is applied to reveal the commitment $\beta_i = PK_i{}^{r_i}.g^{n_{\pi(i)}}$. The commitment β_i is published.

The initialization block, which acts as the genesis block for the chain, is used to initialize the DL in the front-end of the PVPBC voting system. This block does not contain votes. All of the election information, for example, the election public key, the set of valid choices the voters can choose, are included. As a result, a DL is linked to a specific election. As the system parameters are integrated into the DL, potential disagreement over the system parameters is prevented. To allow trustees to join the permissioned network and to participate in maintaining the DL, security credentials are generated for them. A tuple of terms is posted to the DL for each voter V_i, before the voting phase starts, in the following manner:

$$(vk_i \ , \ PK_i \ , \ \{g^{n_{\pi(i)}}\}_{PK_T} \ , \ \beta_i). \tag{2}$$

Election-key generation: In the PVPBC front-end system, tellers are responsible for generating the threshold election key, PK_T. This is the same in the Selene protocol. The election public key, PK_T, is created in a distributed way by the election tellers using the key-generation protocol proposed by Pedersen in [33]. In this protocol, every teller T_j has a share $s_j \in Z_q$ of a secret S.

As the values $PK_j = g^{s_j}$ are made public, every teller is committed to these values. The encryption key, PK_T, for the election is computed using

$$PK_T = \Pi_{j=1}^{n} PK_j, \tag{3}$$

where n is the number of tellers. All tellers are sent the calculated encryption public key, PK_T. It is not possible for a single teller to recover the secret key, $S = \sum_{j=1}^{n} s_j$. The authentication server, which is run by the EA, runs the key-generation algorithm of the digital signature scheme to produce the authentication server's public/private (verification/signing) keys. The primary action that takes place during the election phase, e.g., voting, is now described.

6.2. Election Phase

Voting: After gaining access to the voting site, the voter engages with the voting page, which is part of the front-end system, to cast their ballot. The PVPBC front-end system encrypts the cast ballot using the election key, and then signs the encrypted ballot using the voter's signing key. Then, the front-end system submits the ballot, V_i, to the authentication server at the EA using an authenticated communications channel.

If a ballot is received by the authentication server, and it has the correct format –the ballot is tagged with the correct election identifier–, an acknowledgement is sent as a response by the EA, which consists of a signature applied to the ballot V_i. If the ballot does not conform to the correct format, the EA does not output anything. If the voter or front end tries to vote more than once and an acknowledgement has already been issued by the EA, then the EA reissues the initial acknowledgement and does not consider the new vote. After this set of events, the encrypted vote is published to the permissioned DL alongside the encrypted tracking number, voter's public key, and trapdoor commitment. This is the public record of votes received in encrypted form. This means the signed encrypted vote is created at the vote time, and posted on the WBB when it is cast. Carrying out elections using this process results in the provision of public, real-time information regarding the turnout for the election. The voter is not involved with the voter-side cryptographic actions of the Selene protocol in the voting phase. Instead, the voter's task during the voting phase is to create and cast a ballot in the usual way.

Voting-phase technical details: After casting the ballot, the front-end system carries out several steps to produce the final result. For each cast ballot:

1. The ballot is encrypted and signed, obtaining

$$V_i = sign_{v_i}(\{Vote_i\}_{PK_T}). \tag{4}$$

2. A non-interactive zero-knowledge proof of the knowledge of the plaintext is created, Π_i, and it is attached to the encrypted ballot. In addition, this proof also

includes the voter's verification key, aiming to ensure the proof is valid only for this verification key.

$$V_i = sign_{v_i}(\{Vote_i\}_{PK_T}), (\Pi_i). \tag{5}$$

3. The front end, V_i, submits vi's ballot to the authentication server, EA, using an authenticated channel alongside the voter's VID. If the ballot is correctly formed, then the EA signs the ballot and sends the ballot back to the front-end system.
4. The signed encrypted ballot and the NIZKP are posted on the ledger alongside the previously published encrypted tracking number, tracking number commitment, and voter's identity (PK_i). This process is defined mathematically in

$$(vk_i, PK_i, \{g^{n_{\pi(i)}}\}_{PK_T}, \beta_i, sign_{v_i}(\{Vote\}_{PK_T}), \Pi_i). \tag{6}$$

This is the only information published on the ledger. It is important to note that the standard voting information collected by the election system is not published.

6.3. Post-Election Phase

At the end of the voting phase, as in the Selene protocol, the front-end system starts the mix and decryption process. During this process, each encrypted vote and its corresponding encrypted tracking number are mixed, shuffled, and decrypted by the election tellers. At the end of this process, tracking numbers alongside plaintext votes are posted to the permissioned ledger.

After the votes and tracking numbers have been published, the front-end sends α_i and the corresponding VID to the EA, which, in turn, forwards it to the TTP. The TTP uses the VID to locate the identity of registered voters on the authentication blockchain in order to send their α_i term via e-mail. In response to receiving the α term, the voter can access their tracking number via a call to the front-end system to apply their trapdoor key to (α_i, β_i). This enables them to identify their vote on the ledger.

After the election finishes, the front-end system carries out two steps to produce the final result.

Mix and decryption: For each voter, V_i, the encrypted tracking number and encrypted vote are extracted to give a pair of the form:

$$(\{g^{n_{\pi(i)}}\}_{PK_T}, \{vote_i\}_{PK_T}). \tag{7}$$

This pair is put through a verifiable shuffle performed by the mix-nets [34], and then a threshold set of tellers perform a verifiable decryption of these shuffled pairs. Proof of shuffling and correct decryption are uploaded on the WBB. After decryption, the pairs, $(n_{\pi(i)}, vote_i)$, are published on the WBB.

Tracking-number retrieval: Once the grace period has ended, each voter, V_i, is sent the α_i described in Section 6.1. The received α_i can be combined with the β_i term to reconstruct the tracking number encrypted under the voter's public key PK_i.

$$(\alpha_i, \beta_i) = \{g^{n_{\pi(i)}}\}_{PK_i} \tag{8}$$

The voter, V_i, can request that the front-end system use their secret key to decrypt and retrieve the $g^{n_{\pi(i)}}$, and thus obtains the tracker $n_{\pi(i)}$, which they can then use to look up their vote on the ledger.

7. Security Analysis

A security analysis of the system is now provided. It is organized using the headings: eligibility, privacy, ballot privacy, integrity, and, finally, fairness. Finally, we provide some examples of malicious attacks from which e-voting systems suffer, and analyze the resilience of PVPBC with regard to these attack types.

7.1. Security Attributes

We investigate the strengths and weaknesses of PVPBC with respect to key security attributes.

7.1.1. Eligibility

In order for voters to take part in the election, they need to provide a valid VID. Each valid VID needs to reference a valid capability access token in the authentication DLT. The TTP only provides access tokens to authenticated voters that were included in the list of eligible voters, which was compiled during the initialisation phase, and to authenticated voters that have not previously requested to vote. This means only eligible voters can vote. It also means that they can acquire only one eligibility token, and, therefore, they can only cast one valid vote. The eligibility property is "invalidated" when an eligible voter tries to vote more than once. Invalidation is not possible because the TTP changes the flag field of the tuple to "voted" in the DLT after successful authentication, as part of the voter-authentication phase. Therefore, the voter-authentication request will be rejected when the submitted VID references a profile on the authentication DLT which is flagged "voted".

7.1.2. Privacy

We analyze privacy using two subcategories "Voter Privacy" and "Ballot Privacy".

In terms of voter privacy, the front-end and election provider has no access to the voter identities beyond pseudonyms (VID: unique key allocated by TTP). Voters only submit the VID to the front-end as a proof of identity during the authentication phase. They only need to reveal their real identity to the TTP during the registration phase. Apart from the TTP, no party within the proposed system is able to link a VID to its individual voter.

Regarding ballot privacy, the PVPBC voting system guarantees that no party, at any point of the election, is able to reveal how a particular voter voted. While the front-end is collecting votes, the only link between the voter and their vote in plaintext is the VID key which acts as a pseudonymous identity. Mapping a VID key to the corresponding voter can only be carried out by the TTP, which is responsible for registering eligible voters. Therefore, ballot privacy is maintained with respect to the PVPBC voting system in three ways: the front-end has no access to the voters' true identities; election service providers have no access to both voters and votes; and, finally, the TTP has no access to the plaintext votes linked to their VIDs.

Similar to the Selene protocol, the front end provides the assurance that the published verifiability information does not compromise the privacy of the vote, provided that the requirement that each teller node is independent is met. This is maintained by the front end.

7.1.3. Integrity

The PVPBC-voting-system design ensures that even an insider at the EA or an attacker gaining access to the vote cannot invisibly change votes, as any change can be detected by a voter performing a verification. Consequently, the PVPBC voting system no longer requires the election provider to be trusted to ensure election integrity. This is because it can be verified independently. Use of DLT trustees means that the authority alone cannot change the commitments and verifiability parameters, because collusion between a majority threshold of trustees would be required.

7.1.4. Fairness

The PVPBC voting system guarantees that the election result will not be known during the election period. Therefore, voters cannot be influenced. This property is achieved by separating the election phase from the post-election phase. During the election phase, cast votes are published to the DLT in an encrypted form alongside commitments and encrypted tracking numbers. The ballot is only mixed, shuffled, and decrypted during the post-election phase by a set of tellers.

7.1.5. Verifiability

The PVPBC voting system supports individual and universal verifiability features because it incorporates the Selene voting scheme in its design. Specifically, the proposed system enables universal verifiability because all of the required information is published on the ledger. This allows every independent person to verify cryptographic operations performed at every stage of the election. Some examples of the types of verifications performed are listed. They include verifying the requirements that the tracker numbers are unique, and obtaining proof that shuffling and decryption have all been carried out correctly. Individual verifiability is achieved in the proposed system because voters are able to verify their vote on the DLT after receiving their α term and by reconstructing their tracking number. However, this does not offer the same level of individual verifiability provided by the Selene protocol. This is because the front end has control over the voter's private key. This is in contrast to the Selene protocol where voters control their private key. Consequently, they control the α term and are able to generate a fake α. We believe that sacrificing this Selene protocol feature in the PVPBC voting system is a feature worth sacrificing, given that it helps to preserve the voter usability.

7.1.6. Usability

The PVPBC voting system supports voter usability while preserving privacy and verifiability. Specifically, PVPBC preserves voters' privacy and verifiability without requiring any complex actions to be performed by the voter during the pre-election, election, and post election phases. During the pre-election phase, a voter receives a VID from the TTP to be submitted to the PVPBC's frontend. Once the valid VID is submitted to the frontend, the voter is allowed to access the candidates list and cast her vote. All the voter's cryptographic primitives are managed by the front end (e.g. voter-key generation, and vote encryption, etc). During the post-election phase, the voter will receive the α term which helps in reconstructing the tracking number. The tracking-number reconstruction is performed by the front end, meaning the voter does not perform any complex operation to locate the tracking number associated with her vote. To conclude, voter usability is fulfilled across all election phases without affecting the verifiability and privacy features.

7.2. Malicious-Attack Analysis

E-voting systems are subject to several forms of attack which target the availability, integrity, and anonymity of elections. We discuss and provide a qualitative analysis of the susceptibility of PVPBC to these classes of attack.

7.2.1. Malware

Malware, which includes Trojan horses, worms, spyware, and ransomware, typically targets the integrity of an election by preventing the voter's vote from being recorded as intended [35,36]. Distributed denial-of-service (DDOS) attacks can have a significant negative impact on network routing and performance metrics [37]. A DDOS attack aims to disrupt the voting process by slowing down communication in the network, including vote casting, tallying, and auditing. Regarding PVPBC's resilience to these attacks, vote integrity is achieved as the vote is cast through the front-end system which is controlled and run by the election authority as well as the trustees. We assume that there is a secure channel between the voter and the front-end which reduces the likelihood that adversaries can intercept the votes and alter them during the election phase. Regarding intrusions that intensify DDOS attacks, PVPBC offers a high level of resilience as a result of its use of permissioned blockchain. A consequence of this design choice is that participants are pre-defined and authenticated before joining the network. In addition, every DL node maintains a copy of the voting information (e.g., encrypted votes and verification proofs, etc). This helps PVPBC to stay alive; consequently, it is available for voters if a particular node fails due to a DDOS attack. When the node goes live again, it synchronises with other nodes and updates its DL ledger. In summary, PVPBC achieves a high level of resilience due to its use of blockchain technology.

7.2.2. Cross-Site-Scripting (XSS) Attacks

XSS attacks normally target e-voting systems at the application level. The reason for this is that the application has direct interactions with the voter [38]. As a countermeasure for this type of attack, security measures need to be considered at the application level. In PVPBC, insertion of the voter's contributed content is not allowed at the front-end. Consequently, there is no facility for the voters and candidates to perform editing. In addition, the front-end is designed so that voters' entries are checked and validated against the eligible entry format. This ensures non-persistent XSS attacks are avoided.

7.2.3. Collusion between DLT Participants

We adopted a practical Byzantine fault tolerance (PBFT) consensus protocol which supports e-voting resilience and trust [39]. A PBFT consensus ensures that the system stays available and accessible by voters even if a node goes offline due to a malicious attack. This is an advantage in comparison to e-voting systems which adopted a centralised infrastructure, which is vulnerable to a single point of failure. An additional strength of PVPBC is that we designed the PBFT so that the collusion between PVPBC blockchain members is very challenging. In other words, PBFT prevents the system from reaching an agreement if there are faulty or malicious nodes. To reach an agreement in PBFT, nodes vote on the validity of the information (transactions/blocks). This process provides safety because breaching the data integrity would require $(n-1)/3$ faulty nodes out of a total of n, the honest nodes, to collaborate. The PBFT protocol ensures resilience by including the election authority and other trusted parties as known participants in the permissioned group. They cannot compromise the integrity of the ledger unless the number of participating collaborators surpasses a threshold.

7.2.4. Broken Authentication

Attackers try to gain unauthorised access to online applications, in particular e-voting applications [40]. The possibility of successfully obtaining unauthorised access to e-voting systems during the election is a very serious issue which compromises the integrity of the election. In PVPBC, an authentication protocol based on blockchain is implemented to ensure valid and efficient authentication is carried out. The authentication process is maintained by the election authority in collaboration with the TTP by making use of a capability token which is verified and stored on the blockchain. As blockchain offers an immutable history, it is very challenging to forge the capability access token.

8. Performance Analysis

In this section, we investigate the performance of the PVPBC voting system. To carry out this investigation, we implemented a prototype system. Development of this prototype is initially described. We evaluate the gas and time required to deploy and register a smart contract as part of the registerVoter() function. We then evaluate the gas and time associated with the voter-authorisation phase, where the voter submits an access request to the front-end system. We provide a measurement of the authorisation time, which is the time that elapses between when an access request is submitted to the front-end system and when the request is acted upon. We investigate the impact of different voter population sizes on the authorisation time. The access-request transaction latency within the authentication DLT network is also investigated.

8.1. Prototype System: Voter Registration and Authentication Scheme

We evaluated the PVPBC voting system by implementing a proof-of-concept (PoC) prototype using a Quorum blockchain network. The implementation was divided into three phases: Quorum-network configuration, smart-contract development and deployment, and front-end application development using the REST APIs. For the PoC implementation of the PVPBC voting system, we built REST APIs using Node.js and Express. These APIs allowed us to access and to interact with the blockchain and the functions defined in the smart contract from the front-end system. We used Postman, which is an API development tool, which allowed us to send requests to the Rest APIs and to receive responses. Regarding the TTP and the front-end blockchains, a Byzantine fault tolerance consensus was implemented. The block time was maintained at the default value of 50 ms. We developed all smart-contract code in Solidity using the remix IDE, which allowed us to write, to test, and to debug our smart-contract transactions before deploying the contract on the blockchain network. Finally, we used web3.js and Node.js to interact with the Quorum blockchain and smart contract from the client side. We continue by describing the functionalities implemented in more detail. We then describe the performance evaluation carried out for each function. Finally, we evaluate the voting phase. The goal of this final experiment was to verify how the cost of casting a ballot varied as a function of the number of candidates.

Voter registration: The voter-registration process is described in Algorithm 1. The registerVoter() function in the register-smart-contracts component is used to register a voter on the blockchain. During voter registration, the TTP calls the smart-contract registrar to verify that the provided voter's ID ($voter_{id}$) is part of the whitelist. If the check is successfully passed, the voter is allowed to register themselves for the election ($Access_v$). The TTP handles the registration transaction, which adds the voter's information to the authentication blockchain. Subsequently, an access token is placed on the blockchain. The voter's access token ($Token_v$) is entered by the TTP responsible for pushing the contract to the blockchain. It is then stored in the contract state (smartContract).

Algorithm 1: Voter registration.

```
Input: $voter_{id}$
Output: $VID_v$,$invalid()$
$Access_v = false;$
$candidates = candidates;$
if $(voter_{id} \in candidates)$ then
    $Access_v = true;$
    $Token_v = (timestamp, expDate, Access_v);$
    smartContract= update($Token_v$);
    $VID_v$= address(smartContract)
    return $VID_v;$
else
    return $invalid(Access_v);$
end
```

Time and gas cost: In order to determine the number of units of gas required by PVPBC to perform voter registration, we measured the gas required to deploy the smart contract as part of voter registration for 10 contracts. We repeated each experiment 30 times for each contract, and report the average time in seconds and cost for each contract. The results are tabulated in Table 2 for each contract. The average, of the average time required for each contract, is 34 s and the average gas cost for all contracts is 181000.

Authorisation: During the voter-authorisation phase, the voter submits an access request to the front-end system. This process is described in Algorithm 2. When an access request is initiated using the voter's VID (VID_v), the EA initially checks whether or not the token data associated with the user's address exists in the local database ($localDatabase$). If the search for token data fails, the EA interacts with the authorisation contract (querySmartContract(VIDv)), which fetches the token data from the TTP DLT network by calling an exposed-contract method and saving the token data to the local database. After retrieving the access token, the accessRequest() checks the current capability status of the token, such as the initialized, isValid, issuedate, and expireddate functions. If the status of any token is not valid, the authorisation process stops and a deny-access request is sent back to the subject. The EA goes through all the access rules specified in the retrieved token to decide whether to grant the access to the ballot, or not (e.g., Flag 'voted'). The process checks whether or not the request made by the voter (REST-ful method) to access the ballot matches the access rules in the retrieved access token.

Table 2. Voter registration: The average contract deployment time for 10 contracts, where the average computed for 30 experiments is given and the cost of gas is reported. The average time for deployment over all trials is 34 s and the associated average cost is 181000.

Contract	Time (s)	Cost (Gas)
1	32	182000
2	35	173000
3	33	185000
4	34	180000
5	35	182000
6	35	179000
7	34	184000
8	33	186000
9	35	175000
10	34	184000
Average	34	181000

Algorithm 2: Authorise Access (Function accessRequest()).

```
Input: $VID_v$
Output: $token_{Access}$
Search ($VID_v$);
if ($VID_v \in localDatabase$) then
    $extractToken = token_v$;
else
    extractToken=querySmartContract($VID_v$);
end
verifyTokenStatus(extractToken);
if isValid() then
    EvaluateAccessRight();
    return $token_{Access}$;
else
    Deny();
end
```

Time and gas cost: We measured the gas required to deploy the authorisation contract and the associated time. The experiments were repeated 30 times for each contract, and the results obtained are given in Table 3. The average time reported is 33 s, which is similar to the average time report for voter registration; however, the average gas cost reported is significantly larger, e.g., 549719.

Authorisation time: The duration between when an access request is submitted to the front-end system and when the request is granted or acted upon is called the authorisation time. To measure the authorisation time, we used the ConsenSys Quorum blockchain to emulate the authentication DLT. For the blockchain private network, we implemented three nodes, which were distributed over three Linux machines, to mimic the TTP nodes. We developed a PoC front-end system which allows the voter to submit the VID to the EA.

Table 3. Voter authentication: The average deployment time for 10 contracts and the associated costs are given for 30 experiments. The average of these contract deployment times is 33 s and the average cost is 549719.

Contract	Time (s)	Cost (Gas)
1	49	582033
2	25	539234
3	30	524892
4	35	489234
5	28	547834
6	20	542934
7	27	593485
8	42	534584
9	39	558394
10	35	584574
Average	33	549719

To evaluate the authorisation time, we conducted an experiment which measured the time that elapsed between when the voter access request was made and when the access request was granted. Specifically, we registered voters with the TTP using the registerVoter() function. Access tokens and VID's were created and placed on the DLT. The next step started when VIDs were sent at the same time, T_1, via the front-end application. The experiment used the accessRequest() function in the smart-contract authorisation. This function used VID to fetch the associated token data from the TTP DLT, and it checked the current capability status of the token. We measured the time, T_2, which was when the access token was retrieved and access was granted. The authorisation time, T_A, was measured as the time difference between T_2 and T_1,

$$T_A = T_2 - T_1. \tag{9}$$

In order to determine the impact of different voter population sizes, we ran the experiment several times and increased the voter population for each experiment. The voter population sizes used were 100, 150, 200 , 250, 300, 350, 400, 450, 500 voters. Figure 8 illustrates the average authentication time for each voter population size. Figure 8 provides evidence that the authorisation time increases linearly as a function of the voter population size. For example, the average latency associated with 100 voters obtaining access to the election process is on average 20 s. The average latency increases approximately linearly. When the voter population size is 500 voters, the recorded authentication time latency is approximately 80 s. This linear increase suggests that the voter authorisation time scales well over population sizes of 100 to 500 voters.

The access-request-transaction latency within the authentication DLT network is an important performance metric. The transaction latency in the blockchain environment is the time taken to complete information (transaction) verification. When a node receives a transaction, it verifies whether it is valid or not. If the transaction is valid, the node forwards it to its neighbours. Alternatively, invalid transactions are discarded. We conducted the following experiment to characterise this performance metric. The latency incurred when a valid authentication transaction is created and sent in the DLT network is measured. We also recorded the time at which the access token is retrieved. The goal was to understand the time required to send and receive a response to an access request, which was sent by the EA. We repeated the experiment 30 times for each function call using one client system. Figure 9 plots the latency incurred by this process.

The access-request transaction incurs an average latency of 6.275 ms in Figure 9. It is important to note that the block time of Quorum affects the value of the transaction latency in the system. The block time is defined as the time it takes to add a new block to the blockchain. We employed the default block time of 50 ms, which was set for the

RAFT consensus in this experiment. However, the research of [41] has shown that an increase in block time and transaction rate increases the latency of transactions in the system. This change is because transactions will have to wait longer before they are added to a block, thereby increasing the latency. Current blockchain consensus protocols are not well-adapted for e-voting systems, considering the fact that the existing proof- and voting-based consensus mechanisms may affect the performance of the e-voting systems due to latency and transaction throughput challenges. In future work, we will investigate and design a blockchain consensus protocol suitable for e-voting systems, where transactions are validated with lower latency to ensure fair and efficient performance without compromising security. A related line of attack could also involve adapting modern machine-learning approaches that estimate variation in propagation times [27], improving monitoring [42] and deep-learning classification schemes [29] that aim to perform better utilization of the underlying network resources.

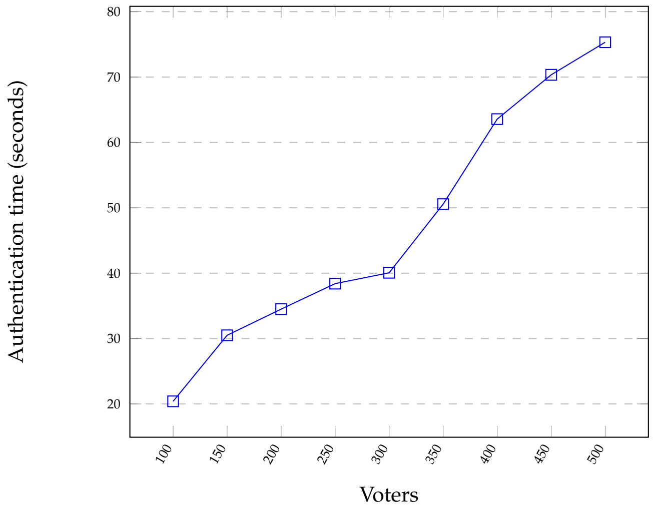

Figure 8. The voter authorisation time is plotted as a function of the number of voters. The authorisation time is an approximately linear function of the voter population size, when it is the range 100 to 500 voters, which underlines the scalability of PVPBC.

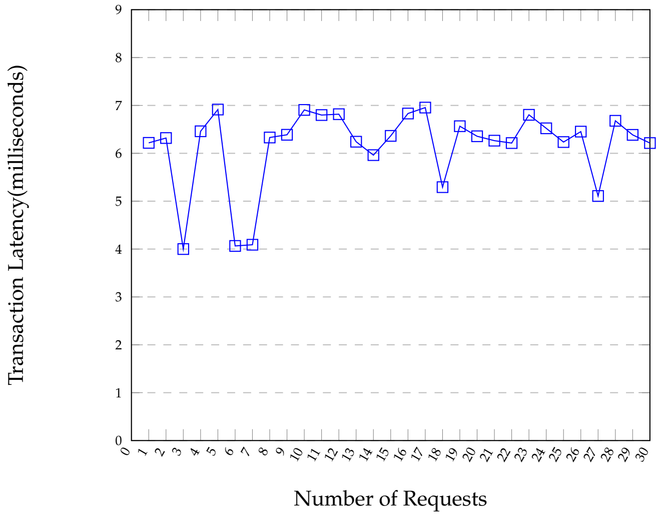

Figure 9. The time required to send and receive a response to an access request is illustrated as a function of the number of requests. The access-request transaction incurs an average latency of 6.275 ms.

8.2. Voting-Phase Performance Evaluation

We deployed our voting-phase implementation, which is described in Section 5.2, on the Quorum private blockchain platform in order to mimic its operation on a real-world, production network. We recruited different numbers of candidates to contest the election, in order to evaluate implementation. The number of candidates was increased from 1 to 7. In this experiment, the blockchain was used as a ballot box during the election phase to store the signed encrypted ballots as well as the tracker commitment (Beta). The goal of these experiments was to verify how the cost for casting a ballot varied with different numbers of candidates. Figures 10 and 11 show the average gas-consumption cost (resp. cost in GBP) for casting an encrypted ballot and tracker commitment based on different numbers of candidates. In this experiment, the cost was calculated in GBP. Based on the data points in Figure 11, it is reasonable to suggest that the costs in GBP increases approximately linearly as the number of candidates increases from 1 to 7. When there are 7 candidates, the cost is GBP 0.6. The cost for 2 candidates is approximately GBP 0.22 . The linear relationship described here to characterise the cost in GBP for casting and storing an encrypted ballot alongside tracker commitment as a function of the number of candidates is an appealing property of PVPBC.

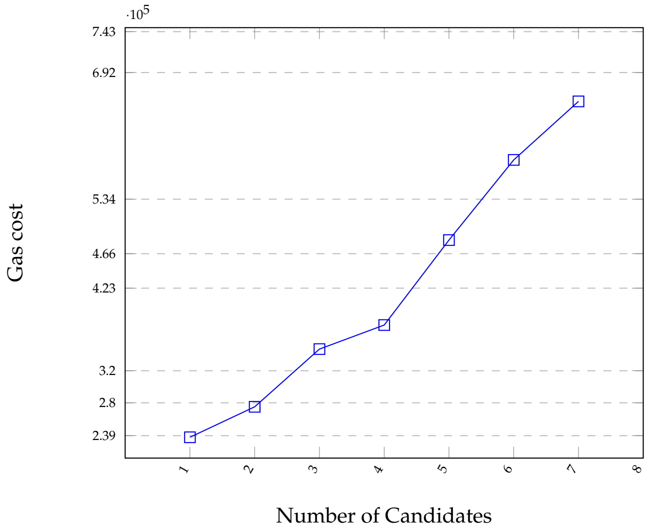

Figure 10. Gas cost for casting and storing an encrypted ballot alongside tracker commitment.

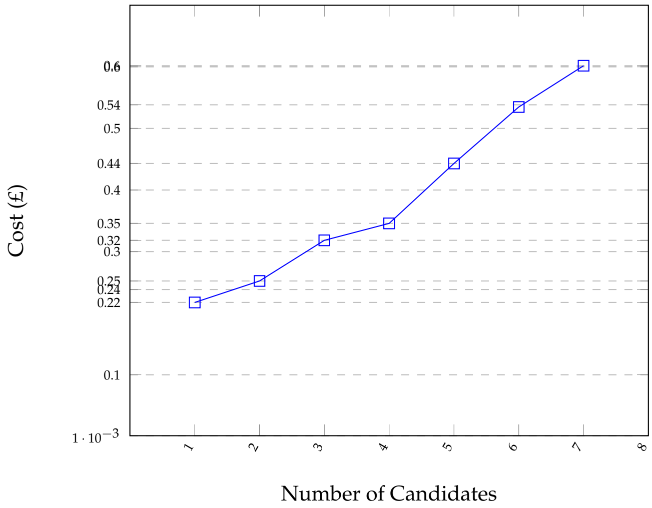

Figure 11. Cost in GBP for casting and storing an encrypted ballot alongside tracker commitment.

9. Conclusions

We proposed an e-voting system based on DLT called PVPBC. The primary advantage of this system is that it preserves voter privacy and verifiability. It achieves this by using DLT to ensure revocable anonymity, and to preserve voter privacy. An additional feature of PVPBC is that it uses advanced cryptographic primitives to support ballot privacy. Voters' verifiability is ensured by using the Selene voting scheme. To demonstrate that the provision of these features is achieved by PVPBC, we provided a detailed security analysis of PVPBC to establish how it preserves privacy, security, verifiability, integrity, fairness, and eligibility. Empirical evidence of the performance characteristics of PVPBC was then provided. We evaluated its performance on a prototype system, by measuring the time and cost required at the registration phase, authentication phase, and voting phase. In summary, the average time required by PVPBC to perform voter registration was 34 s and the associated average cost of gas was 181000 for each contract. During the voter-authorisation phase, the average time to deploy contracts was 33 s and the average cost of gas was 549719. The scalability of the system was then investigated by examining the average authorisation time as the voter population size increased. We provided evidence that the authorisation time was an approximately linear function of the voter population size over the range of population sizes examined. We also reported that access-request transactions incurred an average latency of 6.275 s. The inherent linear relation between various performance metrics and the number of voters and candidates is an appealing property of PVPBC. For example, we suggested that a linear relationship described the cost in GBP for casting and storing an encrypted ballot alongside tracker commitment as a function of the number of candidates. In summation, PVPBC delivers the following key features: privacy, verifiability, integrity, and, finally, eligibility and initial results suggest that the system is scalable.

Author Contributions: M.S.: Conceptualization; Methodology; Software; Validation; Investigation; Resources; Data curation; Writing—original draft preparation; Visualization; Project administration; Writing—review and editing. R.d.F.: Investigation, Conceptualization, Funding acquisition, Project administration Data curation, Writing—original draft preparation, Writing—review & editing. A.M.: Visualization, Data curation, Resources, Writing—original draft preparation, Funding acquisition. All authors have read and agreed to the published version of the manuscript.

Funding: This paper has emanated from research supported in part by a Grant from Science Foundation Ireland under Grant numbers 13/RC/2077_P2 and 15/SIRG/3459.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Tarasov, P.; Tewari, H. The future of e-voting. *IADIS Int. J. Comput. Sci. Inf. Syst.* **2017**, *12*, 148–165.
2. Jafar, U.; Aziz, M.J.A.; Shukur, Z. Blockchain for electronic voting system—Review and open research challenges. *Sensors* **2021**, *21*, 5874. [CrossRef] [PubMed]
3. Sallal, M.; Schneider, S.; Casey, M.; Dragan, C.; Dupressoir, F.; Riley, L.; Treharne, H.; Wadsworth, J.; Wright, P. VMV: Augmenting an internet voting system with Selene verifiability. *arXiv* **2019**, arXiv:1912.00288.
4. Kho, Y.X.; Heng, S.H.; Chin, J.J. A Review of Cryptographic Electronic Voting. *Symmetry* **2022**, *14*, 858. [CrossRef]
5. Mursi, M.F.; Assassa, G.M.; Abdelhafez, A.; Samra, K.M.A. On the development of electronic voting: A survey. *Int. J. Comput. Appl.* **2013**, *61*, 1–11.
6. Arapinis, M.; Kashefi, E.; Lamprou, N.; Pappa, A. A comprehensive analysis of quantum e-voting protocols. *arXiv* **2018**. arXiv:1810.05083.
7. Ryan, P.Y.; Rønne, P.B.; Iovino, V. Selene: Voting with transparent verifiability and coercion-mitigation. In *Financial Cryptography and Data Security: FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, 26 February 2016, Revised Selected Papers 20;* Springer: Berlin/Heidelberg, Germany, 2016; pp. 176–192.
8. Cruz, J.P.; Kaji, Y. E-voting system based on the bitcoin protocol and blind signatures. *IPSJ Trans. Math. Model. Its Appl.* **2016**, *10*, 14–22.
9. Kardaş, S.; Kiraz, M.S.; Bingöl, M.A.; Birinci, F. Norwegian internet voting protocol revisited: Ballot box and receipt generator are allowed to collude. *Secur. Commun. Netw.* **2016**, *9*, 5051–5063. [CrossRef]

10. Basin, D.; Radomirović, S.; Schmid, L. Dispute resolution in voting. In Proceedings of the 2020 IEEE 33rd Computer Security Foundations Symposium (CSF), IEEE, Boston, MA, USA, 22–26 June 2020.
11. Adida, B. Helios: Web-based open-audit voting. In *Proceedings of the 17th conference on Security symposium (SS'08)*; USENIX Association: Berkeley, CA, USA, 2008; pp. 335–348.
12. Chaum, D.; Ryan, P.Y.A.; Schneider, S. A practical voterverifiable election scheme. In Proceedings of the Computer Security—ESORICS 2005, 10th European Symposium on Research in Computer Security, Milan, Italy, 12–14 September 2005; pp. 118–139.
13. Ben-Nun, J.; Fahri, N.; Llewellyn, M.; Riva, B.; Rosen, A.; Ta-Shma, A.; Wikström, D. A new implementation of a dual (paper and cryptographic) voting system. In Proceedings of the 5th International Conference on Electronic Voting, (EVOTE), Bregenz, Austria, 11–14 July 2012.
14. Carback, R.; Chaum, D.; Clark, J.; Conway, J.; Essex, A.; Herrnson, P.S.; Mayberry, T.; Popoveniuc, S.; Rivest, R.L.; Shen, E.; et al. Scantegrity II Municipal Election at Takoma Park: The First E2E Binding Governmental Election with Ballot Privacy. In Proceedings of the 19th USENIX Security Symposium, Washington, DC, USA, 11–13 August 2010.
15. Cortier, V.; Fuchsbauer, G.; Galindo, D. BeleniosRF: A Strongly Receipt-Free Electronic Voting Scheme. *IACR Cryptol. EPrint Arch.* **2015**, *2015*, 629.
16. Clarkson, M.R.; Chong, S.; Myers, A.C. Civitas: Toward a secure voting system. In Proceedings of the IEEE Symposium on Security and Privacy (S&P 2008), Oakland, CA, USA, 18–21 May 2008; pp. 354–368.
17. Benaloh, J. Simple verifiable elections. In Proceedings of the 2006 USENIX/ACCURATE Electronic Voting Technology Workshop, Vancouver, BC, Canada, 1 August 2006.
18. Ryan, P.Y.A.; Teague, V. Pretty good democracy. In Proceedings of the Security Protocols XVII, 17th International Workshop, Cambridge, UK, 1–3 April 2009; Revised Selected Papers, pp. 111–130.
19. Lee, K.; James, J.I.; Ejeta, T.G.; Kim, H.J. Electronic voting service using block-chain. *J. Digit. Forensics Secur. Law* **2016**, *11*, 8. [CrossRef]
20. Meter, C. Design of Distributed Voting Systems. *arXiv* **2017**, arXiv:1702.02566.
21. Bistarelli, S.; Mantilacci, M.; Santancini, P.; Santini, F. An end-to-end voting-system based on bitcoin. In *Symposium on Applied Computing*; ACM: Rochester, NY, USA, 2017.
22. Faour, N. Transparent Voting Platform Based on Permissioned Blockchain. *arXiv* **2018**, arXiv:1802.10134.
23. Benabdallah, A.; Audras, A.; Coudert, L.; Madhoun, N.E.; Badra, M. Analysis of Blockchain Solutions for E-Voting: A Systematic Literature Review. *IEEE Access* **2022**, *10*, 70746–70759. [CrossRef]
24. Leng, J.; Zhou, M.; Zhao, J.L.; Huang, Y.; Bian, Y. Blockchain Security: A Survey of Techniques and Research Directions. *IEEE Trans. Serv. Comput.* **2022**, *15*, 2490–2510. [CrossRef]
25. Sallal, M.; de Fréin, R.; Malik, A.; Aziz, B. An empirical comparison of the security and performance characteristics of topology formation algorithms for Bitcoin networks. *Array* **2022**, *15*, 100221. [CrossRef]
26. Sallal, M.F. Evaluation of Security and Performance of Clustering in the Bitcoin Network, with the Aim of Improving the Consistency of the Blockchain. Doctoral Dissertation, University of Portsmouth, Portsmouth, UK, 2018.
27. de Fréin, R.; Izima, O.; Malik, A. Detecting Network State in the Presence of Varying Levels of Congestion. In Proceedings of the 2021 IEEE 31st International Workshop on Machine Learning for Signal Processing (MLSP), Gold Coast, Australia, 25–28 October 2021; pp. 1–6. [CrossRef]
28. Izima, O.; de Fréin, R.; Malik, A. A Survey of Machine Learning Techniques for Video Quality Prediction from Quality of Delivery Metrics. *Electronics* **2021**, *10*, 2851. [CrossRef]
29. Malik, A.; de Fréin, R.; Al-Zeyadi, M.; Andreu-Perez, J. Intelligent SDN Traffic Classification Using Deep Learning: Deep-SDN. In Proceedings of the 2020 2nd International Conference on Computer Communication and the Internet (ICCCI), Nagoya, Japan, 26–29 June 2020; pp. 184–189. [CrossRef]
30. Sako, K.; Kilian, J. Receipt-free mix-type voting scheme. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Saint-Malo, France, 21–25 May 1995; Springer: Berlin/Heidelberg, Germany, 1995; pp. 393–403.
31. Sallal, M.; Owenson, G.; Salman, D.; Adda, M. Security and performance evaluation of master node protocol based reputation blockchain in the bitcoin network. *Blockchain Res. Appl.* **2022**, *3*, 100048. [CrossRef]
32. Sallal, M.; Owenson, G.; Adda, M. Bitcoin network measurements for simulation validation and parametrisation. In Proceedings of the 11th International Network Conference, Frankfurt, Germany, 19–21 July 2016.
33. Pedersen, T.P. A threshold cryptosystem without a trusted party. In *Workshop on the Theory and Application of of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 1991; pp. 522–526.
34. Wikström, D. *User Manual for the Verificatum Mix-Net Version 1.4. 0*; Verificatum AB: Stockholm, Sweden, 2013.
35. Estehghari, S.; Desmedt, Y. Exploiting the Client Vulnerabilities in Internet E-voting Systems: Hacking Helios 2.0 as an Example. *EVT/WOTE* **2010**, *10*, 1–9.
36. Iqbal, R.; Butt, T.A.; Afzaal, M.; Salah, K. Trust management in social internet of vehicles: Factors, challenges, blockchain, and fog solutions. *Int. J. Distrib. Sens. Netw.* **2019**, *15*, 1550147719825820. [CrossRef]
37. Abuidris, Y.; Kumar, R.; Yang, T.; Onginjo, J. Secure large-scale E-voting system based on blockchain contract using a hybrid consensus model combined with sharding. *Etri J.* **2021**, *43*, 357–370. [CrossRef]

38. Riadi, I.; Raharja, P.A. Vulnerability analysis of E-voting application using open web application security project (OWASP) framework. *Int. J. Adv. Comput. Sci. Appl.* **2019**, *10*. [CrossRef]
39. Feng, L.; Zhang, H.; Chen, Y.; Lou, L. Scalable dynamic multi-agent practical byzantine fault-tolerant consensus in permissioned blockchain. *Appl. Sci.* **2018**, *8*, 1919. [CrossRef]
40. Sohoel, H.; Jaatun, M.G.; Boyd, C. OWASP Top 10-Do Startups Care? In Proceedings of the 2018 International Conference on Cyber Security and Protection of Digital Services (Cyber Security), IEEE, Scotland, UK, 11–12 June 2018; pp. 1–8.
41. Baliga, A.; Subhod, I.; Kamat, P.; Chatterjee, S. Performance evaluation of the quorum blockchain platform. *arXiv* **2018**, arXiv:1809.03421.
42. de Fréin, R. State Acquisition in Computer Networks. In Proceedings of the 2018 IFIP Networking Conference (IFIP Networking) and Workshops, Zurich, Switzerland, 14–16 May 2018; pp. 1–9. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.