DABSTERS: a Privacy Preserving e-Voting Protocol for Permissioned Blockchain

Marwa Chaieb[1], Mirko Koscina[2], Souheib Yousfi[3], Pascal Lafourcade[4] and Riadh Robbana[3]

LIPSIC, Faculty of Sciences of Tunis, University Tunis El-Manar, Tunis, Tunisia[1]
Département d'Informatique, École normale supérieure, Paris, France[2]
LIPSIC, INSAT, University of Carthage, Tunis, Tunisia[3]
LIMOS, University Clermont Auvergne, CNRS UMR6158, Aubiére, France[4]

Abstract. With the immutability property and decentralized architecture, Blockchain technology is considered as a revolution for several topics. For electronic voting, it can be used to ensure voter privacy, the integrity of votes, and the verifiability of vote results. More precisely permissioned Blockchains could be the solution for many of the e-voting issues. In this paper, we start by evaluating some of the existing Blockchain-based e-voting systems and analyze their drawbacks. We then propose a fully-decentralized e-voting system based on permissioned Blockchain. Called DABSTERS, our protocol uses a blinded signature consensus algorithm to preserve voters privacy. This ensures several security properties and aims at achieving a balance between voter privacy and election transparency. Furthermore, we formally prove the security of our protocol by using the automated verification tool, ProVerif, with the Applied Pi-Calculus modeling language.
Keywords: Permissioned Blockchain, Electronic voting, Blind Signature, Formal verification, Applied Pi-Calculus, ProVerif.

1 Introduction

Voting is the cornerstone of a democratic country. The list of security properties that must respect a secure voting protocol includes the following features. **Eligibility:** only registered voters can vote and only one vote per voter is counted. If the voter is allowed to vote more than once, the most recent ballot will be tallied and all others must be discarded. **Individual verifiability:** the voter him/herself must be able to verify that his/her ballot was cast as intended and counted as cast. **Universal verifiability:** after the tallying process, the results are published and must be verifiable by everybody. **Vote-privacy:** the connection between a voter and his/her vote must not be reconstructable without his/her help. **Receipt-freeness:** a voter cannot prove to a potential coercer that he/she voted in a particular way. **Coercion resistance:** even when a voter interacts with a coercer during the voting process, the coercer will be not sure whether the voter obeyed their demand or not. **Integrity:** ballots are not altered or deleted during any step of the election. **Fairness:** no partial results are published before tallying has ended, otherwise voters may be influenced by these results and vote differently. **Robustness:** the system should be able to tolerate some faulty votes. **Vote-and-go:** a voter does not need to wait for the end of the voting phase or trigger the tallying phase. **Voting policy:**

specify if a voter has the right to vote more than once or he/she has not the right to change his/her opinion once he/she casted a vote.

Traditionally, during an election, the voter goes to a polling station and makes his/her choice in an anonymous manner, without any external influence. To perform the tally, we need to trust a central authority. From this comes the risk of electoral fraud. The tallying authority has the possibility to falsify votes and thus to elect a candidate who should not be elected. It is also possible for the registration authority to allow ineligible voters to vote. Hence, voting becomes useless and we notice a decrease in voter turnout in elections. Decentralized systems can be a good alternative to traditional voting since we need a secure, verifiable and privacy preserving e-voting systems for our elections. Blockchain is a distributed ledger that operates without the need to a trusted party. Expanding e-voting into Blockchain technology could be the solution to alleviate the present issues in voting.

Due to the proliferation of Blockchain implementations, the European Blockchain Observatory and Forum has published a technical report [14] where it recommends the use of private or permissioned Blockchains for sensitive data storage, which is the architecture implemented in an e-voting system. In this Blockchain architecture, the user credentials are generated by a Certificate Authority (CA). Hence, the users must be enrolled into the system through the CA before joining the network. This model is suitable for an e-voting system because the user management can rely on the Blockchain platform, due to their formal enrolling process. The advantage of having a minimum level of trust through our knowing the participants is that we can achieve security for the Blockchain replication process by using Byzantine Agreement as a consensus mechanism. Although permissioned Blockchains have several features suitable for services that involve sensitive data, such as user personal information, they have drawbacks related to transactions and user linkability. This is due to the fact that each user credential, public key pair and certificates, are issued for specific users that were previously enrolled in the CA. In order to overcome this drawback, we use, in this paper, the Okamoto-Schnorr blind signature scheme to sign the transactions without linking the user to it. This model allows validating transactions without exposing the user's identification, and therefore maintaining the privacy of the votes.

Related Work: In the last few decades, a considerable number of Blockchain-based e-voting protocols have been proposed to address the security issues of traditional voting protocols. Due to the limitation on the number of pages, we give a brief overview of some of these systems and evaluate their security in Table 1, in which we use the following abreviations[1].

- *Open Vote Network (OVN)* [15]: It is a self-tallying, boardroom scale e-voting protocol implemented as a smart contract in Ethereum. This protocol guarantees voter's privacy and removes the need to trust the tallying authorities whether to ensure the anonymity of voters or to guarantee the verifiability of elections. However, it suffers from several security issues. For example, it supports only elections with two options (yes or no) and with a maximum of 50 voters due to the mathematical tools that they used and to the gas limit for blocks imposed by Ethereum. Additionally, this protocol does not provide any mechanism to ensure coercion resistance

[1] TCA: Trusted Central Authority; SV: Single Vote; MV: Multiple Votes.

and needs to trust the election administrator to ensure voter's eligibility. Open Vote Network is not resistant to the misbehavior of a dishonest miner who can invalidate the election by modifying voters' transactions before storing them on blocks. Dishonest voter can also invalidate the election by sending an invalid vote or by abstaining during the voting phase.

- *E-Voting with Blockchain: An E-Voting Protocol with Decentralization and Voter Privacy (EVPDVP)* [10]: Implemented on a private network that uses the Ethereum Blockchain API, this protocol uses the blind signature to ensure voters privacy. It needs a central authority (CA) as a trusted party to ensure voters eligibility and allow voters to change and update their votes. To ensure fairness, voters include in their ballots a digital commitment of their choices instead of the real identity of the chosen candidate. To tally ballots, voters must broadcast to the network a ballot opening message during the counting phase.
- *Verify-Your-Vote: A Verifiable Blockchain-based Online Voting Protocol (VYV)* [7]: It is an online e-voting protocol that uses Ethereum Blockchain as a bulletin board. It is based on a variety of cryptographic primitives, namely Elliptic Curve Cryptography [9], pairings [4,19] and Identity Based Encryption [5]. The combination of security properties in this protocol has numerous advantages. It ensures voter's privacy because the Blockchain is characterized by the anonymity of its transactions. It also ensures fairness, individual and universal verifiability because the ballot structure includes counter-values, which serve as receipts for voters, and homomorphism of pairings. However, the registration phase of this protocol is centralized. A unique authority, which is the registration agent, is responsible for verifying the eligibility of voters and registering them. A second problem is inherent in the use of Ethereum because each transaction sent by the protocol entities in the Blockchain passes through miners who validate it, put it in the current block and execute the consensus algorithm. Any dishonest miner in the election Blockchain can modify transactions before storing them on blocks. Additionally, this protocol is not coercion resistent.
- *TIVI* [21]: It is a commercial online voting solution based on biometric authentication, designed by the company Smartmatic. It checks the elector's identity via a selfie using facial recognition technology. TIVI ensures the secrecy of votes so long as the encryption remains uncompromised. It provides also voters' privacy thanks to its mixing phase and offers the possibility to follow votes by the mean of a QR code stored during voting phase and checked later via a smartphone application. However, this system does not provide any mechanism to protect voters from coercion or to ensure receipt-freeness. Additionally, TIVI uses the Ethereum Blockchain as a ballot box so it is not resistant to misbehaving miners that could invalidate the election by modifying votes before storing them on the election Blockchain.
- *Follow My Vote (FMV)* [8]: It is a commercial online voting protocol that uses the Ethereum Blockchain as a ballot box. A trusted authority authenticates eligible voters and provides them with pass-phrases needed in case of changing their votes in the future. Voters can watch the election progress in real time as votes are cast. It includes an authentication phase which ensures voters' eligibility. It allows voters

to locate their votes, and check that they are both present and correct using their voters' IDs. Nevertheless, this voting system requires a trusted authority to ensure votes confidentiality and hide the correspondence between the voters' real identities and their voting keys. If this authority is corrupted, votes are no longer anonymous. Votes secrecy is not verified by this system because votes are cast without being encrypted. Moreover, the ability to change votes, coupled with the ability to observe the election in real time compromise fairness property. This system is not coercion resistance and is not universally verifiable because we have no way to verify that the votes present in the election final result are cast by eligible voters.

- *BitCongress [11]*: A commercial online voting platform based on a combination of three networks which are: Bitcoin, Counterparty(a decentralized asset creation system and decentralized asset exchange) and a Smart Contract Blockchain. It aims at preventing double voting by using the time stamp system of the Bitcoin Blockchain. This platform does not ensure voters eligibility because it allows any Bitcoin address to register for the election. It performs the tally using, by default, a modified version of Borda count and a Quota Borda system for large scale elections. It ensures individual and universal verifiability but it is not coercion resistent.
- *Platform-independent Secure Blockchain-based Voting System (PSBVS) [22]:* Implemented in the Hyperledger Fabric Blockchain [2], this protocol uses Paillier cryptosystem [18] to encrypt votes before being cast, proof of knowledge to ensure the correctness and consistence of votes, and Short Linkable Ring Signature (SLRS) [3] to guarantee voters privacy. In the other hand, this protocol does not include a registration phase in which we verify, physically or by using biometric techniques, the eligibility of the voter. A voter can register him/herself by simply providing his/her e-mail address, identity number or an invitation URL with a desired password. These mechanisms are not sufficient to verify the eligibility of a voter and information like e-mail address or identity number can be known by people other than the voter him/herself. Also, with reference to the definition of coercion resistance given by Juels et al. [12], this protocol is not coercion resistant. In fact, if a voter gives his/her secret key to a coercer, the coercer can vote in the place of the voter who cannot modify this vote later. We mention here that the coerced voter cannot provide a fake secret key to the coercer because a vote with a fake secret key is rejected by the smart contract.

Contributions: In this paper, we aim at designing a secure online e-voting protocol that addresses the security issues mentioned in the related work section by using the Blockchain technology and a variety of cryptographic primitives. Called DABSTER, our protocol uses a new architecture based on permissioned Blockchain and blind signature. It satisfies the following security properties: eligibility, individual verifiability, universal verifiability, vote-privacy, receipt-freeness, fairness, integrity and robustness. Our contributions can be summarized as follows:

- A new architecture of trust for electronic voting systems. This architecture is based on permissioned Blockchain and on a blind consensus which provides voter's privacy and vote's integrity.
- A secure and fully distributed electronic voting protocol based on our propounded architecture.

	OVN	EVPDVP	VYV	TIVI	FMV	BitCongress	PSBVS	DABSTERS
Eligibility	TCA	TCA	TCA	✓	TCA	X	X	✓
Individual verif	✓	✓	✓	✓	✓	✓	✓	✓
Universal verif	✓	✓	✓	X	X	✓	✓	✓
Vote-Privacy	✓	✓	✓	✓	TCA	✓	✓	✓
Receipt-freeness	X	✓	✓	X	X	X	✓	✓
Coercion resistance	X	X	X	X	X	X	X	X
Fairness	X	✓	✓	✓	X	X	✓	✓
Integrity	✓	✓	X	✓	X	✓	✓	✓
Robustness	X	✓	✓	✓	✓	✓	✓	✓
Vote-and-go	X	X	✓	✓	✓	✓	✓	✓
Voting policy	SV	MV	MV	SV	MV	MV	SV	MV

Table 1: Security evaluation of OVN, EVPDVP, VYV, TIVI, FMV, BitCongress, PSBVS and DABSTERS.

– A detailed security evaluation of the protocol and a formal security proof using the Applied Pi-Calculus modeling language and the automated verification tool ProVerif.

Outline: In the next section, we give an overview of the Byzantine Fault Tolerante (BFT) with blind signature consensus algorithms. Then in Section 3, we describe our proposed e-voting protocol, DABSTERS, and give its different stakeholders and phases as well as the structure of each voter's ballot. In Section 4, we evaluate the security of our protocol using Proverif when it is possible. The conclusion is a summary of DABSTERS and a proposal for ongoing evaluation of its performance.

2 Background

We give a definition of the Okamoto-Schnorr blind signature, before using it in a Byzantine based consensus.

2.1 Blind Signature

Let p and q be two large primes with $q|p-1$. Let G be a cyclic group of prime order q, and g and h be generators of G. Let $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$ be a cryptographic hash function.

Key Generation: Let $(r,s) \xleftarrow{r} \mathbb{Z}_q$ and $y = g^r h^s$ be the A's private and public key, respectively.

Blind signature protocol:
1. A chooses $(t,u) \xleftarrow{r} \mathbb{Z}_q$, computes $a = g^t h^u$, and sends a to the user.
2. The user chooses $(\beta,\gamma,\delta) \xleftarrow{r} \mathbb{Z}_q$ and computes the blind version of a as $\alpha = ag^{-\beta}h^{-\gamma}y^{\delta}$, and $\varepsilon = H(M,\alpha)$. Then calculates $e = \varepsilon - \delta \bmod q$, and sends e to the A.
3. A computes $S = u - es \bmod q$ and $R = t - er \bmod q$, sends (S,R) to the user.
4. The user calculates $\rho = R - \beta \bmod q$ and $\sigma = S - \gamma \bmod q$.

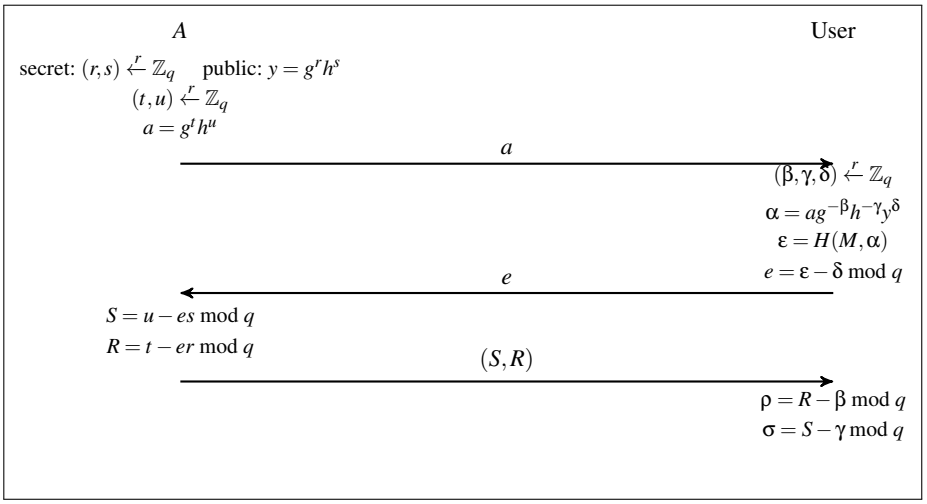

Fig. 1: Okamoto-Schnorr blind signature diagram, where $y \xleftarrow{r} \mathbb{Z}_q$ means that y is randomly chosen in $\mathbb{Z}_q$.

Verification: Given a message $M \in \{0,1\}^*$ and a signature $(\rho,\sigma,\varepsilon)$, we have $\alpha = g^{\rho}h^{\sigma}y^{\varepsilon} \bmod p$.

The Okamoto-Schnorr blind signature scheme is suitable with a private Blockchain architecture due to the blinding process that can be performed by the same authority responsible of the enrollment process (see Figure 1, where the authority A blindly signs a message for the user). We use *BlindSign(M,*$(\beta,\sigma,\gamma),y$) and *VerifyBlindSign(M,*$(\rho,\delta,\varepsilon),y$) to blind sign and to verify the blinded signature, respectively using Okamoto-Schnorr, where M corresponds to the message to be signed, (β,σ,γ) to the secret values randomly chosen, $(\rho,\delta,\varepsilon)$ to the blinded signature; and y to the RA's public key. The result obtained from the function *BlindSign* corresponds to the blinded signature $(\rho,\sigma,\varepsilon)$. On the other hand, the function *VerifyBlindSign* returns a response *valid* or *invalid*.

2.2 BFT based consensus algorithm

Now, considering a permissioned Byzantine Fault Tolerance (BFT) based consensus protocol like the one introduced in Hyperledger Fabric [2]. In this protocol, the digital signature is used as a user authentication method without protecting the user privacy. Hence, for a privacy preserving consensus protocol, we need to add the following properties to the BFT based consensus algorithm:

- Alice sends a newly signed transaction to the registration authority (RA) which is responsible for the enrollment of Alice.
- Alice's signature is validated only by the RA.
- The RA anonymises Alice's identity.
- The RA signs the transaction sent by Alice to the network.
- All the node of the transactions validation process can validate the RA's signature.
- The RA signature cannot be duplicated.

Now, to keep the privacy of the client and peers involved in the transactional process, we need to hide his ID and make his signature blind. However, we do not address the ID hiding process with any particular mechanism. Therefore, we consider that the ID is replaced by a value corresponding to the anonymised user ID, and this process can be performed by using different schemes. As presented in [13], to address the issue related to the digital signature, we replace the signing mechanism used in the original protocol by the Okamoto-Schnorr blind signature scheme [16]. In order to maintain the consistency and liveness that the protocol has, we keep the transactional flow. However, the steps are modified in order to accept the new blind signature scheme to authenticate the clients and peers.

The transactional process based on our BFT consensus algorithm with Blind Signature consists of the following steps:

1. **Initiating Transactions:** The client c_{bc} generates a message M to execute an operation o_{bc} in the network with a blinded signature by using $BlindSign((M,\beta,\sigma,\gamma),y)$.
2. **Transaction Proposal:** The submitting peer sp_{bc} receives the message M coming from the client c_{bc}, validates the client blinded signature by using *VerifyBlindSign(M,*$(\rho,\delta,\varepsilon),y$) and proposes a transaction with the client instruction o_{bc}.
3. **Transaction Endorsement:** The endorser peers ep_{bc} validate the client blinded signature using *VerifyBlindSign(M,*$(\rho,\delta,\varepsilon),y$) and verify if the transaction is valid by simulating the operation o_{bc} using his local version of the Blockchain. Then, the endorser peers generate signed transactions with the result of the validation process and send it to the submitting peer sp_{bc}.
4. **Broadcasting to Consensus:** The submitting peer sp_{bc} collects the endorsement coming from the endorsing peers connected to the network. Once sp_{bc} collects enough valid answers from the endorsing peers, it broadcasts the transaction proposal with the endorsements to the ordering service.
5. **Commitment:** All the transactions are ordered within a block, and are validated with their respective endorsement. Then, the new block is spread through the network to be committed by the peers.

3 Description of DABSTERS

Our protocol is implemented over a new architecture of trust. It is based on a BFT-based consensus protocol [2] and on a blinded signature consensus protocol, called *BlindCons* [13], presented in Section 2. It eliminates the risk of invalidating the election because of dishonest miners who modify the transactions before storing them on blocks. We also propose a distributed enrollment phase to reduce the need to trust election agents and impose the publication of the list of eligible registered voters at the end of the enrollment phase. This list is auditable and verifiable by all parties.

Our scheme unfolds in 5 stages. It starts with an enrollment phase in which registration authorities (RAs) verify the eligibility of voters by verifying the existence of their names and their identity card numbers in a list published beforehand and containing the names of all persons who have the right to vote. Then, all eligible voters are registered and provided with credentials. The enrollment phase is offline. At the end of this phase, RAs construct a list containing the names of all registered eligible voters and their ID

card numbers. This list can be rejected or published on the election Blockchain during the validation phase. Once the list is validated, we move to the third stage which is voting phase. Each eligible voter (V_i) initiates a transaction in which he/she writes his/her encrypted vote, signs the transaction using his/her credential and sends it to the RAs to check his/her signature and blind it. Then, the voter sends the transaction with the blinded signature and his/her anonymous ID (his/her credential) to the consensus peers to be validated and stored in the election Blockchain anonymously. After validating and storing all votes in our Blockchain, tallying authorities (TAs) read these encrypted votes from the network, decrypt them, and proceed to the tally. The final stage is the verification phase. During this phase, voters make sure that their votes have been considered correctly and check the accuracy of the tally. The individual verifiability is ensured due to the structure of our ballots and the universal verification is ensured thanks to the homomorphism property of pairings. Except the enrollment phase, all the phases of our protocol are on-chain. Therefore, we call the BFT based consensus protocol with each transaction initiated by authorities and the BlindCons with each transaction initiated by eligible voters because we do not need to hide the identity of our authorities but we need to ensure voter's privacy. In the following, we give a detailed description of the role of our protocol stakeholders, the structure of our ballot, the different protocol phases and the two consensus.

3.1 Protocol Stakeholders

DABSTERS involves three main entities:

- *Registration authorities (RAs)*: they verify the eligibility of every person wishing to register to the election and provide eligible voters by their credentials which are constructed by cooperation between all RAs.
- *Eligible voters (V)*: every eligible voter (V_i) has the right to vote more than once before the end of the voting phase and only his/her last vote is counted. Voters have the possibility to verify that their votes are cast as intended and counted as cast during the verification phase. Also, they can check the accuracy of the election final result but they are not obliged to participate in the verification phase (they can vote and go).
- *Tallying authorities (TAs)*: the protocol includes as many tallying authorities as candidates. Before the voting phase, they construct n ballots, where n is the number of registered voters. Thus, every voter has a unique ballot which is different from the other ballots. TAs encrypt ballots and send them to voters during the voting phase. They decrypt votes and calculate the election final result during the tallying phase and publish the different values that allow voters to check the accuracy of the count during the verification phase.

DABSTERS also involves observers and election organizers who have the right to host the Blockchain peers to ensure the correctness of the execution of the protocol.

3.2 Ballot Structure:

As illustrated in Figure 2, each ballot is composed of a unique bulletin number BN calculated as follows: $BN = \{g, D\}_{PK_A}$, where g is a generator of an additive cyclic group

G, D is a random number and PK_A is the administrator's public key. It contains also a set of candidates' names $name_j$ and candidates' pseudo IDs, denoted C_j, which are the positions of candidates in the ballot, calculated from an initial order and an offset value. In addition, each ballot includes a set of counter-values $CV_{BN,name_j,k}$ that are receipts for each voter. They are calculated using the following formula: $CV_{BN,name_j,k} = e(Q_{name_j}, S_k \cdot Q_{BN})$; where $e(.,.)$ is the pairing function, S_k is the secret key of the tallying authority TA_k, $Q_{name_j} = H_1(name_j)$ and $Q_{BN} = H_1(BN)$ are two points of the elliptic curve E.

Ballot number BN			
Pseudo "ID C_j"	**Candidate's** "name $name_j$"	**Choice**	**Counter-value** "$CV_{BN,name_j,k}$"
0	Paul	☐	$CV_{BN,name_0,0}$
1	Nico	☐	$CV_{BN,name_1,1}$
2	Joel	☐	$CV_{BN,name_2,2}$

Fig. 2: Ballot structure[17].

3.3 Protocol Stages

Our solution includes the following phases:

Enrollment Phase: Every person who has the right to vote and desires to do so, physically goes to the nearest registration station. He/she provides his/her national identity card to the RAs, who verify his/her eligibility by checking if his/her name and ID card number exists in a list, previously published, contains all persons that are able to participate in the election. If he/she is an eligible voter, the RAs save the number of his/her ID card and provide him with a credential that allows him to participate in the voting process. Voters' credentials are calculated using elliptic curve cryptography and have this form:

$Credential_{V_i} = S_M \cdot H_1(ID_{V_i})$ where:

- $S_M = S_1 \cdot S_2 \ldots, S_R$ is a secret master key calculated by cooperation between all RAs. Each registration authority participates with its secret fragment S_r; $r \in \{1 \ldots R\}$,
- H_1 is an hash function defined as follows: $H_1 : \{0,1\}^* \rightarrow G_1$; G_1 an additive cyclic group of order prime number q,
- ID_{V_i} is the number of the ID card of the voter V_i.

Validation Phase: After registering all eligible voters, RAs create a list containing the names and the identity card numbers of all registered voters. This list should be viewable and verifiable by voters, election organizers and observers. Thus, RAs send this list in a transaction on our election Blockchain. This transaction passes through the five steps of the BFT based consensus protocol to be validated if the list is correct or rejected if the list contains names of ineligible voters.

- **Step1: Transaction initiation.** RAs generate the list of eligible voters to be validated by the network. The list is sent to a submitter peer. In the case of an offline or misbehaving submitter peer, RAs send the transaction to the next submitter peer. This step is illustrated by Figure 3.
  - ID_{RA} is the ID of the registration authorities,
  - $Write(List)$ is the operation invoked by the RAs to be executed by the network. It consists of writing the list of eligible voters and their ID card numbers in the Blockchain,

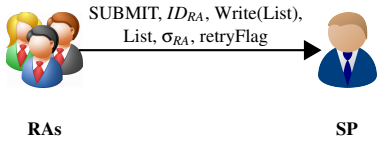

Fig. 3: Step1: Transaction initiation.

- *List* is the payload of the submitted transaction, which is the list of registered voters to be published on the Blockchain,
- σ_{RA} is the signature of the registration authorities,
- $retryFlag$ is a boolean variable to identify whether to retry the submission of the transaction in case of the transaction fails.

– **Step2: Transaction proposal.** The submitter peer receives the transaction and verifies the RAs signature. Then prepares a transaction proposal to be sent to the endorsing peers. Endorsing peers are composed of some voters, election organizers and observers who desire to host the Blockchain peers. This step is described in Figure 4.

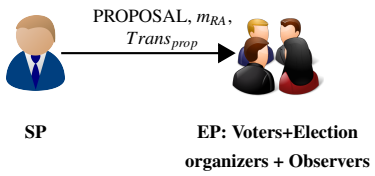

Fig. 4: Step2: Transaction proposal.

- $m_{RA} = (ID_{RA}, Write(List), List, \sigma_{RA})$
- $Trans_{prop} = (SP, Write(List), List, StateUpdate, VerDep)$:
  - *StateUpdate* corresponds to the state machine after simulate locally the operation coming in *Write(List)*.
  - *VerDep* is the version dependency associated to the variable to be created or modified. It is used to keep the consistency of the variables across the different machine state version.

– **Step3: Transaction endorsement.** Each endorser peer verifies the signature of the registration authorities σ_{RA} coming in m_{RA} and checks that the list of eligible voters in m_{RA} and $Trans_{prop}$ is the same. Then, each endorser verifies the eligibility of all names and ID card numbers included in the list. If they are all valid, the endorser peer generates *a transaction valid message* to be sent to the submitter peer (Figure 5). But if the list includes names of ineligible voters, the endorser peer generates *a transaction invalid message* (Figure 6).
  - Tx_{ID} is the transaction ID,
  - σ_{EP} is the signature of the endorser peer.
  - $Error$: can has the following values:
    - INCORRECT-STATE: when the endorser tries to validate the transaction with a different local version of the Blockchain than the one coming in the transaction proposal.

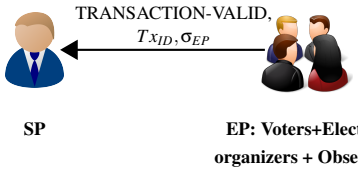

Fig. 5: Step3: Transaction endorsement: valid transaction.

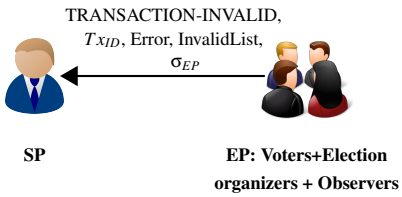

Fig. 6: Step3: Transaction endorsement: invalid transaction.

  - ∗ INCORRECT-VERSION: when the version of the variable where the list will be recorded differs from the one referred in the transaction proposal.
  - ∗ REJECTED: for any other reason.
- InvalidList: is the list of ineligible names that were included in the list sent by the RAs.

– **Step4: Broadcasting to Consensus.** The submitter peer waits for the response from the endorser peers. When it receives enough *Transaction Valid messages* adequately signed, the peer stores the endorsing signatures into packaged called endorsement. Once the transaction is considered endorsed, the peer invokes the consensus services by using *broadcast(blob)*, where *blob = (Trans$_{prop}$, endorsement)* (Figure7).
The number of responses and endorsement to consider the transaction proposal as endorsed is equal to $50\% + 1$ of the total number of endorser peers. If the transaction has failed to collect enough endorsements, it abandons this transaction and notifies the RAs.

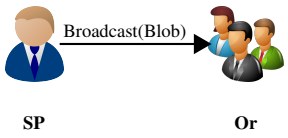

Fig. 7: Step4: Broadcasting to consensus.

– **Step5: Commitment.** Once the submitter peer broadcasts the transaction to consensus, the ordering services put it into the current block, which will be sent to all peers once built. Finally, if the transaction was not validated, the registration authorities are informed by the submitter peer SP.

In the case of an invalid list, the registration authorities have to correct the list and restart the validation phase. We move to the next phase (which is the voting phase) only when we obtain a valid list of registered voters.

Voting Phase: Two entities participate during this phase:

– The tallying authorities who have constructed ballots before the beginning of the voting phase. To construct a ballot, TAs calculate, locally, the unique ballot number $BN = \{g, D\}_{PK_{TA}}$, the offset value $offset = H(g) \ mod \ m$ and the counter-values $CV_{BN,name_j,k} = e(Q_{name_j}, S_k \cdot Q_{BN})$, where g is a generator of G an additive cyclic group of order a prime number, D is a random number, PK_{TA} is TAs' public key, m is the number of candidates, $e(.,.)$ is the pairing function, S_k is the secret key of the tallying authority TA_k, $Q_{name_j} = H_1(name_j)$ and $Q_{BN} = H_1(BN)$ are two points of the elliptic curve E. Then, TAs choose, randomly, a blank ballot for each voter, encrypt it with the voter's public key and transmit it to the corresponding voter via the Blockchain. Ballots are sent encrypted because they contain secret information like the BN, the $offset$ and counter-values. To send encrypted ballots to voters via the Blockchain, TAs interact with the BFT consensus peers. These interactions unfold in five steps, the same steps as those presented in Section 3.3, and described in Figure 8.

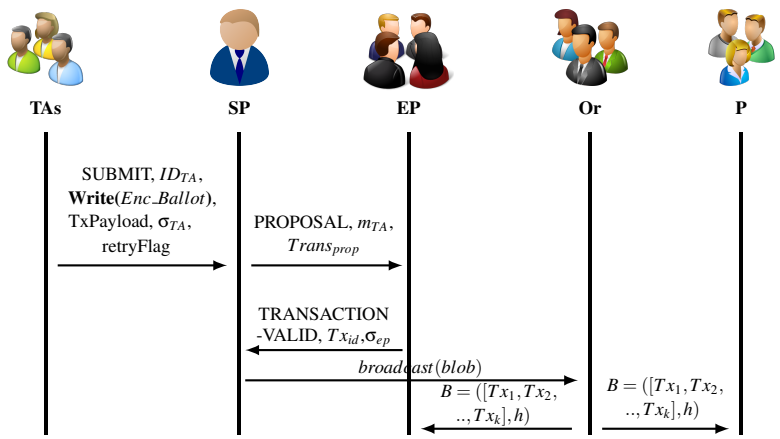

Fig. 8: Interaction between TAs and peers.

1. **Transaction initiation.** TAs initiate a transaction and send it to a submitter peer SP. The transaction contains their ID (ID_{TA}), the list of encrypted ballots, the transaction payload, their signature (σ_{TA}) and the value of the variable retryFlag.
2. **Transaction proposal.** SP verifies the TAs signature and prepares a transaction proposal $Trans_{prop} = (SP, Write(Enc\_Ballot), Enc\_Ballot, stateUpdate, VerDep)$ to be sent to the endorsing peer with the TAs message $m_{TA} = (ID_{TA}, Write(Enc\_Ballot), Enc\_Ballot, \sigma_{TA})$

3. **Transaction endorsement.** EP verifies the σ_{TA} coming in m_{TA}, simulates the transaction proposal and validates that the $stateUpdate$ and $verDep$ are correct. If the validation process is successful, the endorser peer generates a transaction valid message to be sent to the submitter peer.
4. **Broadcasting to consensus.** When the SP receives a number of *Transaction Valid message* equals to $50\% + 1$ of the total number of endorser peers, adequately signed, he stores the endorsing signatures into an endorsement package and invokes the consensus services by using $broadcast(blob)$; where $blob = (Trans_{prop}, endorsement)$.
5. **Commitment.** Ordering services (Or) add the transaction to a block. Once they collect enough endorsed transactions, they broadcast the new block to all other peers. A block has the following form: $B = ([tx_1, tx_2, \ldots, tx_k]; h)$ where h corresponds to the hash value of the previous block.

– Every eligible voter retrieves his/her ballot, decrypts it using his/her secret key and encrypts his/her vote by voting then sends it in a transaction through the Blockchain. To encrypt his/her vote, the voter uses the Identity Based Encryption [5] and encrypts his/her ballot number BN with $Q_{C_j} = H_1(C_j)$ where C_j is the pseudo ID of the chosen candidate. Thus, each encrypted vote has the following form: $Enc\_Vote = \{BN\}_{Q_{C_j}}$.
To be read from the Blockchain or be written on it, voters' transactions pass through the blinded signature consensus. We model in Figure 9 the steps through which a transaction of an eligible voter passes. We take the example of a transaction containing an encrypted vote. During the interactions between TAs and peers, we use the

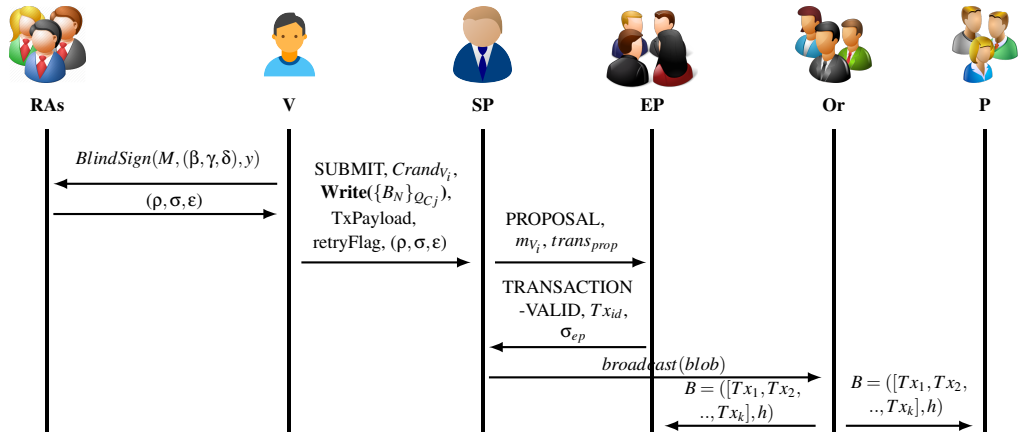

Fig. 9: Interactions between eligible voter and BlindCons peers.

digital signature as user authentication method without protecting the TAs privacy because we do not need to hide the identity of our protocol authorities. However, when it comes to interactions between voters and peers, we need to preserve vot-

ers' privacy by blinding their signatures. The privacy preserving consensus adds two steps:

i) The signature of each eligible voter is blinded automatically after the vote is cast by the function $BlindSign(M,(\beta,\gamma,\delta),PK_{RA})$, where $M = (Credential_{V_i}||Write(Enc\_Vote)||Enc\_Vote||retryFlag)$ is the message to be signed, (β,γ,δ) are secret values randomly chosen by the voter and PK_{RA} is the public key of the RAs.
ii) RAs blind the signature of each eligible voter by providing him the tuple (R,S), allowing the voter to construct his/her blinded signature $(\rho,\sigma,\varepsilon)$ to be used during his/her interactions with the peers.

The other steps are the same as the BFT based consensus, but instead of sending their signatures, the voters send their blinded signatures provided by the RAs.

1. **Initiating transaction:**
   <SUBMIT, $Credential_{V_i}$, $Write(Enc\_Vote)$, Enc_vote, retryFlag, $(\rho,\sigma,\varepsilon)$ >
2. **Transaction Proposal:** <PROPOSAL,m_{V_i},$trans_{prop}$ >
3. **Transaction Endorsement:** < TRANSACTION-VALID, Tx_{id}, σ_{ep} >
4. **Broadcasting to consensus:** broadcast(blob)
5. **Commitment:** $B = ([Tx_1,Tx_2,\ldots,Tx_k],h)$

The voters who intend to verify that their votes were properly counted must memorize the counter-values that correspond to their chosen candidates.

Tallying Phase: After all votes have been cast, TAs proceed to the tally. We have as many TAs as candidates. Each tallying authority TA_k is responsible for counting the number of votes for a specific pseudo ID C_j: for example the first tallying authority TA_1 decrypts, with its secret key $S_1 \cdot Q_{C_1}$, all bulletins that were encrypted with the public key Q_{C_1} (certainly these ballots contain votes for candidates with $C_j = 0$). TA_k starts by initiating a transaction to read encrypted votes from the Blockchain. This transaction passes through the five steps of the BFT based consensus. Then, it decrypts the votes with its secret key S_k that were encrypted with Q_{C_j} in order to reveal the bulletin number BN. Then, it reconstructs the ballot, identifies the chosen candidate, and added to the corresponding counter. At the end of this phase, TA_k publishes the count for each candidate using the following formula: $\sigma_{k,name_j} = \sum_{i=1}^{l_j} S_k \cdot Q_{BN_i}$; Where l_j is the number of votes received by the candidate j, S_k is the private key of the tallying authority k, $Q_{BN_i} = H_1(BN_i)$ and BN_i is the ballot number of the vote i that corresponds to the candidate with name $name_j$.

Verification Phase: This phase allows voters to check that their votes were counted as cast and that the election final result corresponds to the sum of all eligible votes. It includes two sub-phases. During the first one, TAs calculate the list of chosen counter-values from each ballot number and the name of the chosen candidate, and publish this list on the Blockchain. Each eligible voter can read this list and verify the existence of his/her counter-value to be sure that his/her vote was counted correctly. The second sub-phase uses the homomorphism of pairings to check the accuracy of the tally. Using the published counts and the reconstructed counter-values, we can verify that the

announced result corresponds to the sum of all eligible votes, as follows :

$$\prod_{i=1}^{l} CV_{BN_i} = \prod_{k=1}^{m}\prod_{j=1}^{m}\prod_{i=1}^{l_j} CV_{BN_{i,name_j},k} = \prod_{k=1}^{m}\prod_{j=1}^{m}\prod_{i=1}^{l_j} e(Q_{name_j}, S_k \cdot Q_{BN_i})$$
$$= \prod_{k=1}^{m}\prod_{j=1}^{m} e(Q_{name_j}, \sum_{i=1}^{l_j} S_k \cdot Q_{BN_i}) = \prod_{k=1}^{m}\prod_{j=1}^{m} e(Q_{name_j}, \sigma_{k,name_j}) \quad (1)$$

Where $l = \sum_{j=1}^{m} l_j$ is the total number of votes. These equalities use the bilinear property of pairing: $\prod_{i=1}^{l_j} e(Q_{name_j}, S_k \cdot Q_{BN_i}) = e(Q_{name_j}, \sum_{i=1}^{l_j} S_k \cdot Q_{BN_i})$

4 Security Evaluation of DABSTER

Thanks to the use of the BFT based consensus, the BlindCons and a variety of cryptographic primitives, our protocol ensures several security properties. We discuss the security properties ensured by our protocol and prove, formally, that our solution guarantees vote secrecy, vote privacy, and voter's authentication.

4.1 Informal Security Evaluation

We evaluate our protocol according to a list of security properties that must respect a secure and practical voting system.

- **Eligible voter:** The registration and the validation phases of our protocol ensure that only eligible voters participate in the voting process. During the registration phase, RAs verify the identity of each voter via a face to face meeting and only eligible voters are provided with credentials. During the validation phase, RAs send the list of registered voters to the consensus peers, which are composed of voters, election organizers and observers, in order to verify the eligibility of all registered voters and validate or reject this list.
- **Individual verifiability:** This property is ensured by our protocol because our ballot structure includes counter-values. These values serve as receipts for voters and enable them to verify that their votes have been cast as intended without disclosing who they voted for. In fact, counter-values are calculated using the following formula: $CV_{BN,name_j,k} = e(Q_{name\,j}, S_k \cdot Q_{BN})$. Thus, we cannot get the name of the candidate from the value of $CV_{BN,name_j,k}$.
- **Universal verifiability:** From the parameters published by the TAs during the verification phase, everyone can verify the accuracy of the final result by checking the equation (1).
- **Vote-Privacy:** This property is ensured thanks to the BlindCons. Before interacting with the consensus peers, RAs blind the signature of all eligible voters to hide their real identities. Voters' transactions are signed by the blind signature issued by the RAs and not with the voter's signature. Thus voters' identities are kept private and no one can link a vote to a voter.

- **Receipt-freeness:** In our case, a voter cannot find his/her vote from the counter-value $CV_{BN,name_j,k}$ and the other public parameters. He/she cannot therefore prove that he/she voted for a given candidate.
- **Coercion resistance:** Our protocol is not resistant to coercion. A coercer can force a voter to vote for a certain candidate and check his/her submission later using the counter-value.
- **Integrity:** The BFT based consensus and the blind signature algorithm prevent votes from being altered while keeping the voter's secrecy. Each transaction is stored in the Blockchain after being validated by $50\% + 1$ of the endorsing peers. This eliminates the risk of modifying transactions before storing them. We mention here that the BFT consensus is based on the assumption that $2/3$ of the endorsing peers are honest.
- **Fairness:** During the voting phase, each eligible voter encrypts his/her ballot number BN with $Q_{C_j} = H_1(C_j)$ where C_j is the pseudo-ID of the desired candidate. Ballot numbers are secret and candidates' pseudo-IDs do not reflect the real identities of candidates thanks to the offset value, so nobody can identify the chosen candidate from the encrypted vote. Thus, we cannot get partial results before the official count.
- **Robustness:** Our scheme is resistant to the misbehavior of dishonest voters which cannot invalidate the election by casting an invalid vote or by refusing to cast a vote.
- **Vote-and-go:** Our protocol does not need the voter to trigger the tallying phase, they can cast their votes and quit before the voting ends.
- **Voting policy:** DABSTERS gives the possibility to eligible voters to vote more than once and only their last votes are counted. It means that we have a maximum of one vote per voter in the final tally. In fact, every eligible voter has a unique valid credential which is sent with his/her vote in the transaction.

4.2 Formal Security Evaluation

ProVerif is a fully automated and efficient tool to verify security protocols. It is capable of proving reachability properties, correspondence assertions, and observational equivalence. To perform an automated security analysis using this verification tool, we model our protocol in the Applied Pi-Calculus [1] which is a language for modeling and analyzing security protocols. It is a variant of the Pi-Calculus extended with equational theory over terms and functions and provides an intuitive syntax for studying concurrency and process interaction. The Applied Pi-Calculus allows to describe several security goals and to determine whether the protocol meets these goals or not. To describe our protocol with the Applied Pi calculus, we need to define a set of names, a set of variables and a signature that consists of the function symbols which will be used in order to define terms. These function symbols have arities and types. To represent the encryption, decryption, signature, blind signature and hash operations, we use the following function symbols: `pk(skey)`, `aenc(x,pk(skey))`, `adec(x,skey)`, `spk(sskey)`, `sign(x,sskey)`, `checksign(x,spk(sskey))`, `BlindSign(x,smkey)`, `checkBlindSign(x, spk(smkey))`, `H1(x)`. Intuitively, the `pk` function generates the corresponding public key of a given secret key, `aenc` and `adec` stand, respectively, for

asymmetric encryption and asymmetric decryption, `aenc` and `adec` follow this equation: `adec(aenc(x,y),pk(y))=x`. The `spk` function generates the corresponding public key of a given signature secret key, `sign` and `checksign` provide, respectively, the signature of a given message and the verification of the signature. They respect the following equation: `checksign(sign(x,y),spk(y))=x`. `BlindSign` and `checkBlindSign` stand, respectively, for blind sign and check blinded signature, `BlindSign` and `checkBlindSign` follow the equation `checkBlindSign(BlindSign(x,y),spk(y))=x`. We also assume the hash operation which is denoted with the function `H1`.

Because of the limitation on the number of pages, we put all ProVerif codes online[2] and give only the queries, the results of executing these codes, and the time it takes ProVerif to prove the properties in Table 2 (Execution times are expressed in seconds).

Property to evaluate	Description	Result	Exec time
Vote secrecy	To capture the value of a given vote, an attacker has to intercept the values of two parameters: the ballot number *BN* and the pseudo ID of the chosen candidate Cj.	Proved	0.012s
Voter's Authentication	We use correspondence assertion to prove this property.	Proved	0.010s
Vote privacy	To express vote privacy we prove the observational equivalence between two instances of our process that differ only in the choice of candidates.	Proved	0.024s

Table 2: ProVerif results and execution times.

4.3 Blockchain Security Evaluation

DABSTERS Blockchain protocol has the following security properties.

Consistency: A Blockchain protocol achieves consistency if it is capable of ensuring that each valid transaction sent to the network will stay immutable in the Blockchain.

Definition 1 (Consistency). *A Blockchain protocol $\mathbb{P}$ is $T-$consistent if a valid transaction tx is confirmed and stays immutable in the Blockchain after $T-$round of new blocks.*

Theorem 1. *DABSTERS Blockchain protocol is 1-consistent.*

Proof. The consistency is achieved by agreeing on the validity of the transaction through a Byzantine Agreement process. Hence, the probability to not settling it in a new block is negligible if the transaction has at least $50\% + 1$ of valid endorsement and the network have at most $\lfloor \frac{an-1}{3} \rfloor$ out of total n malicious peers, as it has been shown in [6] [14] under the terminology of safeness. The protocol achieves consistency after a new block is created (1-consistency) due to the chain is growing without forks.

[2] `http://sancy.univ-bpclermont.fr/~lafourcade/DABSTERS_FormalVerif/`

Liveness: A consensus protocol ensures liveness if a honest client submits a valid transaction and a new block is generated with the transaction in it. Hence, the protocol must ensure that the Blockchain growths if valid clients generate valid transactions.

Definition 2 (Liveness). *A consensus protocol* $\mathbb{P}$ *ensures* liveness for a Blockchain C *if* $\mathbb{P}$ *ensures that after a period of time* t*, the new version of the Blockchain* C' *is* $C' > C$*, if a valid client* $c_{i_{bc}}$ *has broadcasted a valid transaction* tx_i *during the time* t*.*

Theorem 2. *DABSTERS Blockchain protocol achieves liveness.*

Proof. Our protocol is a BFT-based consensus. Thus, liveness is achieved if after the transaction validation process, the network agrees in new block B with the transactions broadcasted by the clients during a period of time t. Hence, for valid transactions tx_i, where $i \in \mathbb{N}_0$, issued by valid a client c_i during a period of time t, the probability that $C' = C$ is neglected if we have at most $\lfloor \frac{n-1}{3} \rfloor$ out of total n malicious peers [6].

Blindness: We use the definition of *blindness* defined by Schnorr in [20]. A signature is properly blinded if the signer cannot get any information about the signature if the receiver follows the protocol correctly.

Definition 3 (Blind signature). *A signing scheme is* blind *if the signature* $(m, \rho, \sigma, \varepsilon)$ *generated by following correctly the protocol, is statistically independent of the interaction* (a, e, R, S) *with that provides the view to the signer.*

Theorem 3. *Okamoto-Schnorr signature* $(m, \rho, \delta, \varepsilon)$ *is statistically independent to the interaction* (a, e, R, S) *between the authority A and the user.*

Proof. We recall how the protocol works. To generate a blind signature $(m, \rho, \sigma, \varepsilon)$ the user chooses randomly $(\beta, \gamma, \delta) \in \mathbb{Z}_q$ to respond to the commitment a generated by A with the challenge $e = H(m, ag^{\beta}h^{\gamma}y^{\delta}) - \delta \bmod q$. The authority A then sends $(R, S) = (t - er, u - es)$ to finally obtain the signature by calculating $(\rho, \sigma) = (R - \beta, S - \gamma)$. Hence, for the constant interaction (a, e, R, S) and a unique set (β, γ, δ) randomly chosen per signature, we generate a signature $(m, \rho, \delta, \varepsilon) = (m, R - \beta, S - \gamma, e + \gamma)$ that is uniformly distributed over all the signatures of the message m due to the random $(\beta, \gamma, \delta) \leftarrow \mathbb{Z}_q$ [20].

5 Conclusion

We proposed a fully decentralized electronic voting system that combines several security properties. This protocol, called DABSTERS, uses a new architecture that allows enhancement of the security of e-voting systems and guarantees the trustworthiness required by voters and election organizers. DABSTERS is designed to be implemented on private Blockchains and uses a new blinded signature consensus algorithm to guarantee vote integrity and voter's privacy due to the unlinkability property that the blinded signature has. Future work will be dedicated to evaluating the performance and the scalability of DABSTERS.

Acknowledgments This work has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No 826404, Project CUREX.

References

1. Martín Abadi, Bruno Blanchet, and Cédric Fournet. The applied pi calculus: Mobile values, new names, and secure communication. *J. ACM*, 65(1):1:1–1:41, 2018.
2. Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference, EuroSys 2018, Porto, Portugal, April 23-26, 2018*, pages 30:1–30:15. ACM, 2018.
3. Man Ho Au, Sherman S. M. Chow, Willy Susilo, and Patrick P. Tsang. Short linkable ring signatures revisited. In Andrea S. Atzeni and Antonio Lioy, editors, *Public Key Infrastructure, Third European PKI Workshop: Theory and Practice, EuroPKI 2006, Turin, Italy, June 19-20, 2006, Proceedings*, volume 4043 of *Lecture Notes in Computer Science*, pages 101–115. Springer, 2006.
4. Dan Boneh. Pairing-based cryptography: Past, present, and future. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, volume 7658 of *Lecture Notes in Computer Science*, page 1. Springer, 2012.
5. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.
6. Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *Proceedings of the Third USENIX 1999*, 1999.
7. Marwa Chaieb, Souheib Yousfi, Pascal Lafourcade, and Riadh Robbana. Verify-your-vote: A verifiable blockchain-based online voting protocol. In Marinos Themistocleous and Paulo Rupino da Cunha, editors, *Information Systems - 15th European, Mediterranean, and Middle Eastern Conference, EMCIS 2018, Limassol, Cyprus, October 4-5, 2018, Proceedings*, volume 341 of *Lecture Notes in Business Information Processing*, pages 16–30. Springer, 2018.
8. Followmyvote. Follow my vote. `https://followmyvote.com/`, 2012.
9. Darrel Hankerson and Alfred Menezes. Elliptic curve cryptography. In Henk C. A. van Tilborg, editor, *Encyclopedia of Cryptography and Security*. Springer, 2005.
10. Freya Sheer Hardwick, Raja Naeem Akram, and Konstantinos Markantonakis. E-voting with blockchain: An e-voting protocol with decentralisation and voter privacy. *CoRR*, abs/1805.10258, 2018.
11. BIT Congress Inc. Bitcongress. `http://cryptochainuni.com/wp-content/uploads/BitCongress-Whitepaper.pdf`.
12. Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In David Chaum, Markus Jakobsson, Ronald L. Rivest, Peter Y. A. Ryan, Josh Benaloh, Miroslaw Kutylowski, and Ben Adida, editors, *Towards Trustworthy Elections, New Directions in Electronic Voting*, volume 6000 of *Lecture Notes in Computer Science*, pages 37–63. Springer, 2010.
13. Mirko Koscina, Pascal Lafourcade, David Manset, and David Naccache. Blindcons: A consensus algorithm for privacy preserving private blockchains. Technical report, LIMOS, 2018. `http://sancy.univ-bpclermont.fr/~lafourcade/BlindCons.pdf`.

14. Jinyuan Li and David Maziéres. Beyond one-third faulty replicas in byzantine fault tolerant systems. In *NSDI*, 2007.
15. Patrick McCorry, Siamak F. Shahandashti, and Feng Hao. A smart contract for boardroom voting with maximum voter privacy. In *FC 2017*, volume 10322. Springer, 2017.
16. Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *Crypto 92*, pages 31–53. Springer, 1992.
17. S. Narayan P. Udaya and V. Teague. A secure electronic voting scheme using identity based public key cryptography. In *Proceedings of SAR-SSI 2007, Annecy, France, June 12-16, 2007*, 2007.
18. Pascal Paillier. Paillier encryption and signature schemes. In Henk C. A. van Tilborg and Sushil Jajodia, editors, *Encyclopedia of Cryptography and Security, 2nd Ed.*, pages 902–903. Springer, 2011.
19. Francesco Rossi and Giovanni Schmid. Identity-based secure group communications using pairings. *Computer Networks*, 89:32–43, 2015.
20. Claus Peter Schnorr. Security of blind discrete log signatures against interactive attacks. In *International Conference on Information and Communications Security*, pages 1–12. Springer, 2001.
21. Smartmatic. Tivi. `http://www.smartmatic.com/voting/online-voting-tivi/`, 2016.
22. Bin Yu, Joseph K. Liu, Amin Sakzad, Surya Nepal, Ron Steinfeld, Paul Rimba, and Man Ho Au. Platform-independent secure blockchain-based voting system. In Liqun Chen, Mark Manulis, and Steve Schneider, editors, *Information Security - 21st International Conference, ISC 2018, Guildford, UK, September 9-12, 2018, Proceedings*, volume 11060 of *Lecture Notes in Computer Science*, pages 369–386. Springer, 2018.