DABSTERS: Distributed Authorities using Blind Signature To Effect Robust Security in e-voting

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

[1] *Faculty of Sciences of Tunis, University Tunis El-Manar, Tunis, Tunisia*

[2] *Département d'Informatique, École normale supérieure, Paris, France*

[3] *LIP2, National Institute of Applied Science and Technology, University of Carthage, Tunis, Tunisia*

[4] *LIMOS, University Clermont Auvergne, CNRS UMR6158, Aubière, France*

Keywords: E-voting, Blind Signature, Fully-Decentralized, Permissioned Blockchain

Abstract: Creating an online electronic voting system that meets all legal requirements of election organizers and voters has constituted a real challenge for a long period of time. Permissioned Blockchains (also called Private Blockchains) are a cutting-edge invention, introduced as a security breakthrough for many existing and emerging technologies. One potential application of private Blockchain concerns e-voting systems. We propose a fully-decentralized e-voting system based on permissioned Blockchain, called DABSTERS in e-voting. Our system uses a blinded signature consensus algorithm, which is a modified version of Practical Byzantine Fault Tolerance (PBFT), to preserve voter's privacy. Our protocol ensures several security properties: voter's eligibility, vote integrity, vote secrecy, fairness, receipt freeness, individual and universal verifiability.

1 Introduction

Blockchain technology is a distributed, immutable and public ledger that ensures transparency, decentralization, irreversibility, and non-repudiation properties. The first Blockchain implementation is Bitcoin (Nakamoto, 2008), which corresponds to a permissionless architecture, where everybody can anonymously join the network. In 2014, Vitalik Buterin proposed a second permissionless Blockchain implementation called Ethereum (Buterin, 2014). Ethereum combines the traditional idea of the distributed ledger with smart contracts built on the top of a virtual machine. However, the advantages of Blockchain come with significant drawbacks such as an unbearable computational complexity and limited scalability. But its most important weakness is the efficiency of the mechanism used to avoid the misbehavior of dishonest miners. Thus, for every new block, it is required to follow a consensus mechanism to agree on the next block to be appended to the chain, such as Proof-of-Work (PoW) and Proof-of-Stake (PoS) used respectively in Bitcoin and Ethereum. In these consensus mechanisms, miners collect the transactions, validate them and organize them in a block. Once a miner has successfully passed the selection mechanism imposed by the consensus algorithm to propose a new block, the selected miner broadcasts the block to the network to be validated by the rest of miners and then to be added to the chain. Dishonest miners can modify transactions data before storing them on the blocks. This issue gets worse especially if the exchanged data are sensitive or important. It is the case of e-voting systems where exchanged data are votes. Dishonest miners can invalidate elections by modifying these transactions. Moreover, permissionless Blockchains are not suitable to manage tailored business rules, like special data access restrictions and user profiling. These special requirements can be addressed by using permissioned Blockchain. Thus, we can rely on private Blockchain not just for the immutability of the system but also on the users' management for sensitive services like e-voting. Hence, our aim is to design a fully distributed secure architecture for electronic voting systems.

Our contributions are:

- A new architecture of trust for e-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 e-voting protocol, called *DABSTERS in e-voting* for Distributed Authorities using Blind Signature To Effect Robust Security, based on our propounded architecture.

DABSTERS in e-voting satisfies the following security properties. **Eligibility:** Only registered voters can vote, and nobody can submit more votes than allowed. **Fairness:** No preliminary results that could influence other voters' decisions are made available. **Robustness:** The protocol can tolerate misbehaving voters. **Integrity:** Is the assurance of the accuracy and consistency of votes. **Individual Verifiability:** Each voter can check whether his vote was counted correctly. **Universal Verifiability:** Anybody can verify that the announced result corresponds to the sum of all votes. **Vote-Privacy:** The votes are kept private. This can also be modeled as an unlinkability between the voter and his vote. **Receipt-Freeness:** A voter cannot generate a receipt to prove to a third party which candidate has voted for.

Related Work: We study various voting systems based on Blockchain technology and we resume their security properties in Table 1.

Open Vote Network (McCorry et al., 2017): It is a self-tallying, boardroom scale e-voting protocol implemented as a smart contract in Ethereum. Open Vote Network ensures votes privacy since votes are encrypted before being cast. This protocol is self tallying so it is universally verifiable and ensures individual verifiability thanks to the use of Blockchain. 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 and needs to trust the election administrator to ensure voter's eligibility. Open Vote Network is not resistant to the misbehavior of a dishonest miners who can invalidate the election by modifying voters' transactions before storing them on blocks. A dishonest voter can also invalidate the election by sending an invalid vote.

TIVI (Smartmatic, 2016): It is an 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 several security properties such as voters' eligibility since it provides different authentication techniques and votes secrecy 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 (Followmyvote, 2012): It is an 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. Follow My Vote respects a limited number of security properties. 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 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, a coercer can force a voter to vote in a certain way and check his submission later using his pass-phrase. FMV is not universally verifiable because there is no way to verify that the votes present in the election final result are cast by eligible voters.

Verify-Your-Vote: A Verifiable Blockchain-based Online Voting Protocol (Chaieb et al., 2018): It is an online electronic voting protocol that uses Ethereum Blockchain as a bulletin board. It is based on a variety of cryptographic primitives, namely Elliptic Curve Cryptography, pairings and Identity Based Encryption. The combination of security properties in this protocol has numerous advantages. As shown in Table 1, it ensures voter's privacy because the Blockchain is characterized by the anonymity of its transactions. It also ensures fairness, individual and universal verifiability thanks to the ballot structure that includes counter-values and to the homomorphism property of pairings. However, VYV suffers from acknowledged weaknesses where the honesty of human being is involved especially in the registration phase and the architecture of the system. A first problem is that the registration phase is centralized. A unique authority, which is the registration agent, is responsible for verifying the eligibility of voters and registering them. A dishonest agent can register persons who do not have the right to vote. Thus, ineligible voters, who are provided by valid authentication parameters, can access the Blockchain and participate in the election. 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. Again in VYV, any dishonest miner in the election Blockchain can modify transactions before storing them on blocks. This risk constitutes a real issue if dishonest miners modify eligible voters' transactions during the voting phase. These transactions contain encrypted votes. If they change their values, tallying authorities (TAs) will not be able to decrypt them. As a result, TAs count only votes that were not modified by dishonest miners during the voting phase. The other votes will be discarded, thus, some eligible voters may have their choices not counted in the election final tally. Therefore, dishonest miners cause false election result.

DABSTERS in e-voting comes with significant improvements to address the weaknesses of (Chaieb et al., 2018). It proposes a new architecture of trust, based on permissioned Blockchain and blind signature. It also proposes a distributed registration phase and adds a validation phase, where all parties can check the validity of registered voters list and have the right to reject it if they detect a dishonest behavior from the RA.

Outline: In the next section, we give an overview of the PBFT and the blinded signature consensus algorithms. Then in Section 3, we describe our proposed e-voting protocol, DABSTERS in e-voting, give its different entities and phases, and give an overview of our ballot structure. The conclusion is a summary of DABSTERS in e-voting protocol and a proposal for ongoing evaluation of its performance.

2 Background

We give a definition of the Okamoto-Schnorr blind signature (Okamoto, 1992), before using it in a PBFT based consensus.

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 $H : \{0,1\}^* \to \mathbb{Z}_q$ be a cryptographic hash function.

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

Blind signature protocol:

1. A chooses $(t,u) \overset{r}{\leftarrow} \mathbb{Z}_q$, computes $a = g^t h^u$, and sends a to the user.
2. The user chooses $(\beta,\gamma,\delta) \overset{r}{\leftarrow} \mathbb{Z}_q$ 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 $\sigma = S - \gamma \bmod q$.

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. We use *BlindSign(M,*$(\beta,\sigma,\gamma),y$) and *Verify-BlindSign(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*.

PBFT: Now, considering a permissioned Practical Byzantine Fault Tolerance (PBFT) based consensus protocol like the one introduced in Hyperledger Fabric (Androulaki et al., 2018). 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 PBFT based consensus algorithm:

- A voter V_i sends a newly signed transaction to the registration authority (RA) which is responsible for the enrollment of voters.
- V_i's signature is validated only by the RA.
- The RA anonymises the identity of V_i.
- The RA signs the transaction sent by V_i 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 voter and peers involved in the transactional process, we need to hide his/her ID and make his/her signature blind. As presented in (Koscina et al., 2018), 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. 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.

3 Description of DABSTERS in e-voting

Our protocol uses the same ballot structure as (Chaieb et al., 2018), illustrated in Figure 1, and proposes a totally different architecture for the sys-

	VYV	OVN	TIVI	FMV	DABSTERS
Eligibility	Trusted authority	Trusted admin	✓	Trusted authority	✓
Fairness	✓	X	✓	X	✓
Robustness	✓	X	✓	✓	✓
Integrity	X	X	X	X	✓
Individual verifiability	✓	✓	✓	✓	✓
Universal verifiability	✓	✓	X	X	✓
Vote-Privacy	✓	✓	✓	✓	✓
Receipt-freeness	✓	X	X	X	✓
Coercion resistance	X	X	X	X	X

Table 1: Security properties of VYV, OVN, TIVI, FMV and DABSTERS in e-voting.

tem. It is based on a PBFT based consensus protocol (Androulaki et al., 2018) and on a blinded signature consensus protocol, called *BlindCons* (Koscina et al., 2018). 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 during the validation phase. 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. 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 writes his encrypted vote, signs the transaction using his credential and sends it to the RAs to check his signature and blind it. Then, the voter sends the transaction with the blinded signature and his anonymous ID (his credential) to the consensus peers to be validated and stored, anonymously, in the election Blockchain. 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 has been considered correctly and check the accuracy of the tally. The individual verifiability is ensured thanks to the counter-values included in each ballot 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 PBFT based consensus protocol with each transaction initiated by authorities and BlindCons with each transaction initiated by eligible voters because we don't 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 stakeholder, its phases and the two consensus.

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	□	$CV_{BN,name_2,2}$

Figure 1: Ballot structure(P. Udaya and Teague, 2007).

3.1 Protocol Stakeholder

DABSTERS in e-voting involves three main entities:

- *Registration authorities (RAs)*: they verify the eligibility of every person wishing to register to the election and provide only 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 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 to the verification phase (they can vote and go).
- *Tallying authorities (TAs)*: the protocol includes as many tallying authorities as candidates. 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, before the voting phase, encrypt and send them to vot-

ers during the voting phase, 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 in e-voting 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 Protocol Stages

Our solution includes the following phases:

3.2.1 Enrollment Phase

Every person who has the right to vote and desires to do so, physically goes to the nearest registration station. He provides his national identity card to the RAs, who verify his eligibility by checking if his name and ID card number exists in a list, previously published, contains all persons that are able to participate in the election. If he is an eligible voter, the RAs save the number of his 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.

3.2.2 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 PBFT 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 (SP). In the case of an offline or misbehaving SP, RAs send the transaction to the next one. The structure of the message is as follow:
  $< SUBMIT, ID_{RA}, List, TxPayload, \sigma_{RA} >$ where:
  - 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,
  - $List$: 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 RAs,
  - $retryFlag$: is a boolean variable to identify whether to retry the submission of the transaction in case of the transaction fails.
- **Step2: Transaction proposal.** The SP receives the transaction and verifies the RAs' signature. Then prepares a transaction proposal to be sent to the endorsing peers (EP). EP are composed of some voters, election organizers and observers who desire to host the Blockchain peers. Transaction proposal has the following form:
  $< PROPOSAL, m_{RA}, Trans_{prop} >$ where:
  - $m_{RA} = (ID_{RA}, Write(List), List, \sigma_{RA})$
  - $Trans_{prop} = (SP, Write(List), List, StateUpdate, VerDep)$:
    - *StateUpdate* corresponds to the state machine after simulating 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 EP verifies the signature of the RAs 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 EP verifies the eligibility of all names and ID card numbers included in the list. If they are all valid, the EP generates *a transaction valid message* to be sent to the SP. This transaction includes the following values: $< TRANSACTION-VALID$, Tx_{ID}, $\sigma_{EP} >$ where:
  - Tx_{ID} is the transaction ID,
  - σ_{EP} is the signature of the endorser peer.

  But if the list includes names of ineligible voters, the EP generates *a transaction invalid message*, which has the following form: $< TRANSACTION-INVALID$, Tx_{ID}, $Error$, $InvalidList$, $\sigma_{EP} >$ where:
  - *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.
    - 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 SP waits for the response from the EP. 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)*. 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 the enough endorsements, it abandons this transaction and notifies the RAs.
- **Step5: Commitment.** Once the SP broadcasts the transaction to consensus, the ordering services (Or) put it into the current block, which will be sent to all peers once built. Finally, if the transaction was not validated, the RAs are informed by the 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.

3.2.3 Voting Phase

Two entities participate during this phase:

- The TAs 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 $CV_{BN,name_j,k} = e(Q_{namej}, S_k \cdot Q_{BN})$, 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, $Q_{namej} = H_1(name_j)$ 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 PBFT consensus peers. These interactions unfold in five steps, the same steps as those presented in section 3.2.2, and described in Figure 2.
  1. **Transaction initiation.** TAs initiate a transaction and send it to a 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 EP with the TAs message $m_{TA} = (ID_{TA}, Write(Enc\_Ballot), Enc\_Ballot, \sigma_{TA})$
  3. **Transaction endorsement.** EP verifies 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 SP.
  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 $broadcast(blob)$; wher $blob = (Trans_{prop}, endorsement)$.
  5. **Commitment.** Ordering services (Or) add the transaction to a block. Once the Or collects enough endorsed transactions to construct a block, he broadcasts the new block to all other peers. A block has the following form: $B = ([tx_1, tx_2, \dots, tx_k]; h)$ where h corresponds to the hash value of the previous block.
- Every eligible voter retrieves his ballot, decrypts it using his secret key and encrypts his vote by voting then sends it in a transaction through the Blockchain. To encrypt his vote, the voter uses the Identity Based Encryption and encrypts his 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 3 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 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

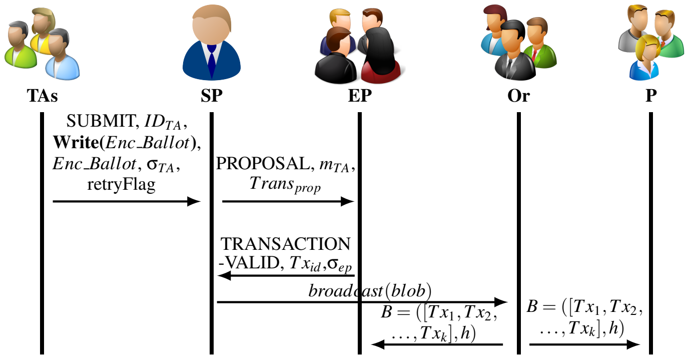
TAs
SP
EP
Or
P
SUBMIT, ID_{TA}, **Write**(Enc_Ballot), Enc_Ballot, σ_{TA}, retryFlag
PROPOSAL, m_{TA}, $Trans_{prop}$
TRANSACTION -VALID, Tx_{id},σ_{ep}
$broadcast(blob)$
$B = ([Tx_1, Tx_2, \ldots, Tx_k], h)$
$B = ([Tx_1, Tx_2, \ldots, Tx_k], h)$

Figure 2: Interaction between TAs and peers.

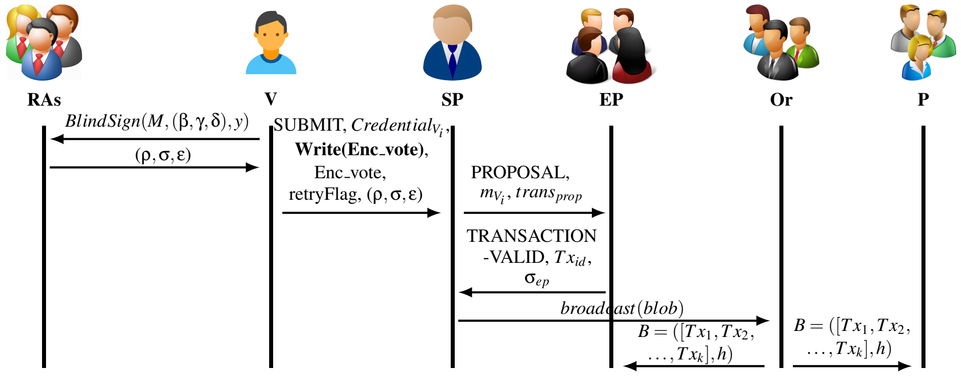
RAs
V
SP
EP
Or
P
$BlindSign(M, (\beta, \gamma, \delta), y)$
$(\rho, \sigma, \varepsilon)$
SUBMIT, $Credential_{V_i}$, **Write(Enc_vote)**, Enc_vote, retryFlag, $(\rho, \sigma, \varepsilon)$
PROPOSAL, m_{V_i}, $trans_{prop}$
TRANSACTION -VALID, Tx_{id}, σ_{ep}
$broadcast(blob)$
$B = ([Tx_1, Tx_2, \ldots, Tx_k], h)$
$B = ([Tx_1, Tx_2, \ldots, Tx_k], h)$

Figure 3: Interactions between eligible voter and BlindCons peers.

and peers, we need to preserve voters' 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) to construct his blinded signature $(\rho, \sigma, \varepsilon)$ to be used during his interactions with the peers.

The other steps are the same as the PBFT 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,$Tx_{id}, \sigma_{ep} >$
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 before casting their votes.

3.2.4 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 PBFT 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 $name_j$.

3.2.5 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 counter-value to be sure that his 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 :

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

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 Conclusion

We proposed a fully decentralized e-voting system that combines several security properties, called DABSTERS in e-voting. It uses a new architecture that allows enhancement of the security of e-voting systems and guarantees the trustworthiness required by voters and election organizers. It 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 this protocol.

REFERENCES

Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., Caro, A. D., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., et al. (2018). 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.

Buterin, V. (2014). A next generation smart contract & decentralized application platform.

Chaieb, M., Yousfi, S., Lafourcade, P., and Robbana, R. (2018). Verify-your-vote: A verifiable blockchain-based online voting protocol. In Themistocleous, M. and da Cunha, P. R., editors, *Information Systems - 15th European, Mediterranean, and Middle Eastern Conference, EMCIS, 2018, Proceedings*, volume 341 of *Lecture Notes in Business Information Processing*, pages 16–30. Springer.

Followmyvote (2012). Follow my vote. `https://followmyvote.com/`.

Koscina, M., Lafourcade, P., Manset, D., and Naccache, D. (2018). Blindcons: A consensus algorithm for privacy preserving private blockchains. Technical report, LIMOS. `http://sancy.univ-bpclermont.fr/~lafourcade/BlindCons.pdf`.

McCorry, P., Shahandashti, S. F., and Hao, F. (2017). A smart contract for boardroom voting with maximum voter privacy. In *FC 2017*, volume 10322. Springer.

Nakamoto, S. (November 2008). Bitcoin: A peer-to-peer electronic cash system.

Okamoto, T. (1992). Provably secure and practical identification schemes and corresponding signature schemes. In *Crypto 92*, pages 31–53. Springer.

P. Udaya, S. N. and Teague, V. (2007). A secure electronic voting scheme using identity based public key cryptography. In *Proceedings of SAR-SSI 2007, Annecy, France, June 12-16, 2007.*

Smartmatic (2016). Tivi. `http://www.smartmatic.com/voting/online-voting-tivi/`.