# An End-to-end Voting-system Based on Bitcoin[*]

Stefano Bistarelli, Marco Mantilacci, Paolo Santancini, Francesco Santini
Dipartimento di Matematica e Informatica, University of Perugia
[bista, mantilacci, santancini, francesco.santini]@dmi.unipg.it

## ABSTRACT

In this work we re-adapt the Bitcoin e-payment system and propose it as a decentralised end-to-end voting platform (from voters to candidates). We describe the main architectural choices behind the implementation, which consists of the pre-voting, voting, and post-voting phases. The resulting implementation is completely decentralised: it is possible to directly cast a vote in the block-chain without any collecting intermediate-level. All the votes can be verified by anyone reading such a public ledger. We also exploit digital asset coins to directly keep track of votes (through the *Open Asset Protocol*), and we show the election cost for *n* voters.

## CCS Concepts

**•Computer systems organization → Peer-to-peer architectures; •Applied computing → Voting/election technologies; •Information systems → Digital cash;**

## Keywords

Electronic voting; Bitcoin; verifiable voting-system.

## 1. INTRODUCTION AND MOTIVATIONS

In short, *electronic voting* [7] (also known as *e-voting*) is voting using electronic systems to aid casting and counting votes. In this work we propose an *end-to-end* (E2E, i.e. from voters to count) *verifiable* system[1] [5] based on the well-known *crypto-currency*, i.e., bitcoins [9] (with sign ฿). In E2E systems, *i)* any voter may check whether her ballot has been correctly included in the electronic ballot box, and *ii)* anyone may determine if all ballots have been correctly counted.

In this work, we take the inspiration from an historically related field, i.e., e-payment systems, and the Bitcoin platform [9] in the specific.[2] Electors anonymously authenticate online (multiple schemes can be used, e.g., *Anonymous Kerberos* [12, 14] or *Blind signature* [4]) and, as a consequence, receive a token to vote (a fraction of a bitcoin) in their *wallet* (see Sec. 2 for the technical details on Bitcoin). Afterwards, the voter can "spend" such a token by transferring it to the address of the desired candidate: this is recorded, once for all, in the Bitcoin *block-chain*, which consists in a public distributed-ledger of sequential transactions: this avoids the need to have a centralised database managed by a (trusted) third party. As it is designed, there is a prohibitively high (computational) cost to attempt to rewrite or alter the block-chain [9, 10]. Finally, the result can be verified by directly counting in the block-chain the tokens paid to each candidate.

The proposed solution is architecturally distributed (Bitcoin is natively peer-to-peer), and no central authority is needed. Moreover, with Bitcoin being used since 2009 by more than 4 million users,[3] such a platform has been already extensively tested in the large: the 24-hour transaction volume is currently 40 million €.[4]

A similar framework is offered by the *Bitcongress.org* project[5], which already proposes a voting-platform based on *Ethereum*[6]. Ethereum is a public block-chain based distributed computing-platform, featuring *smart contract* functionalities.[7] It is based on a cryptocurrency called *ether*. One more platform, still based on Ethereum, is *FollowMyVote*[8], which enhances the anonymity of voters by using elliptic curve cryptography on the stored transactions. Works as [13] or [1] propose a distributed protocol that all the voters need to implement (similarly to lottery protocols): adherence to the protocol is provided through zero-knowledge proofs. In [13], Bitcoin is mainly used as a way to reward or penalise users that respectively behave correctly or incorrectly, since they need to invest some bitcoins before to vote: such an amount can be refunded back to the voter or not, depending on her behaviour.

[*]This work has been partially supported by project "Bitcoin", funded by Banca d'Italia.
[1]One E2E system is e.g. Helios: https://vote.heliosvoting.org.

*SAC 2017,April 03-07, 2017, Marrakech, Morocco*
Copyright 2017 ACM 978-1-4503-4486-9/17/04. . . $15.00
http://dx.doi.org/10.1145/3019612.3019841

[2]In the paper we use "Bitcoin" to refer to the protocol and network, and "bitcoin" to the currency.
[3]https://blockchain.info/charts/my-wallet-n-users.
[4]Reference: https://markets.blockchain.info.
[5]http://www.bitcongress.org.
[6]https://www.ethereum.org.
[7]A Smart contract is computer protocols that facilitates, verifies, or enforces the negotiation or performance of a contract.
[8]https://followmyvote.com.

## 2. BITCOIN

The white-paper on Bitcoin appeared in November 2008 [9], written under the pseudonym of "Satoshi Nakamoto". His invention is an open-source, peer-to-peer digital currency (being electronic, with no physical manifestation). Money transactions do not require a third-party intermediary, such as credit cards. The Bitcoin network is completely decentralised, with all parts of transactions performed by the users of the system. A complete transaction record of every Bitcoin and every Bitcoin user's encrypted identity is maintained on a public ledger. For this reason, Bitcoin transactions are thought to be "pseudonymous".

As represented in Fig. 1, the main actors of the platform are *users*, who own a *wallet* associated with a couple of private/public cryptographic keys. Users use these keys to sign transactions (irreversible by design), which are then broadcast to the Bitcoin peer-to-peer network. Some of the peers in this network are the *miners*, whose task is to update the *block-chain*, a public distributed data-structure that implements the database of every transaction ever executed.

A bitcoin owner transfers the coin to a different owner (e.g., a Buyer to a Seller in Fig. 1 by digitally signing a hash of a previous transaction (proving this owner is in possess of some previously received bitcoins) and the public key of the next owner. The use of multiple inputs corresponds to the use of multiple coins in a cash transaction. A transaction can also have multiple outputs, allowing a owner to make multiple payments at once. A payee can verify the signatures to verify the ownership chain. The sum of inputs can exceed the intended sum of payments, but, as in cash transactions, an additional output is used to return change back to the payer. Transaction fees are voluntary on the part of the payer, and they just represent an incentive for miners.

Given such a framework, a problem would be that the payee cannot verify that one of the owners is not double-spending the same money. An easy but weak solution is to introduce a trusted central-authority, checking every transaction for double spending. Moreover, the only way to confirm the presence of a transaction in the history is to be aware of all the past transactions. Without a trusted party, transactions must be publicly announced, and all participants have to agree on a single history of the order in which transactions are received. The payee needs a proof that the majority of network nodes agree that transaction money is spent for the first time.

*Mining* is a record-keeping service that compute such proof-of-work. Miners keep the block-chain consistent, complete, and unalterable: they repeatedly verify and collect newly broadcast transactions into a new group of transactions, called a *block*. A new block contains information that chains it to the previous block, that is a hash of the previous block. Each block is also computationally impractical to modify, since every block after it would also have to be regenerated. In order to be accepted by the rest of the Bitcoin network, a new block must contain a so-called *proof-of-work*. The proof-of-work (whose computation is the main task of miners) consists in finding a "nonce". This value corresponds to a 32-bit field (in each block), which when inserted into the current block (header) makes the hash be below the current *target*. The target is a 256-bit number (extremely large) that all Bitcoin

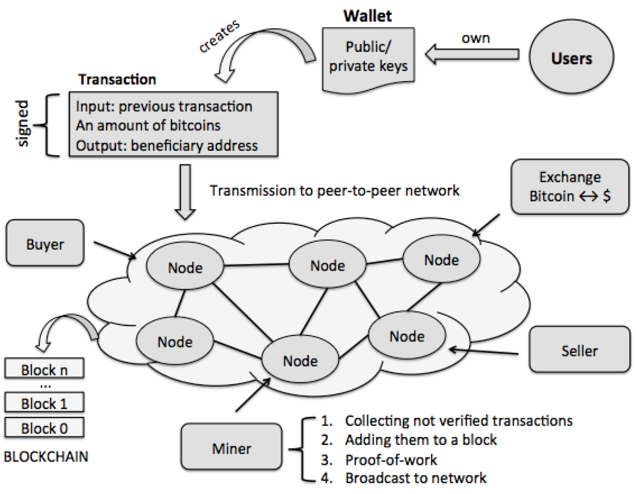

**Figure 1: The Bitcoin peer-to-peer network in a glimpse.**

clients share. The most widely used scheme for hashing is SHA-256, similarly to Adam Back's *Hashcash* [2]. This proof is easy to verify, but extremely time-consuming to generate: the demanded work is exponential in the number of "zero bits" required in the head of the block-hash, but it can be verified by other peers in a single hash. Note that such amount of effort is continuously adjusted with the aim of keeping the average time between new blocks at 10 minutes.

A chain is valid if all of the blocks and transactions within it are confirmed, and only if it starts with the *genesis block*. For any block in the valid chain, there is only one path to the genesis block. Climbing up from the genesis block, however, there can be forks. One-block forks are sometimes created when two blocks are created just a few seconds apart; in such cases, generating nodes build onto whichever one of the blocks they have received first. Nodes always consider the longest chain to be the correct one and will keep working on extending it. If two nodes broadcast different versions of the next block simultaneously, some nodes may receive one or the other first. In that case, they work on the first one they received, but save the other branch in case it becomes longer. The tie will be broken when the next proof-of-work is found and one branch becomes longer; nodes working on the other branch will then switch to the longer one.

## 3. VOTING WITH BITCOIN

In this section we describe how to use the Bitcoin platform to implement an E2E voting-system. According to the high-level models of election systems [3], and to many proposals in the literature as the taxonomy in [11] recollects, e-voting has three phases with two/three sub-phases each:

1) Pre-voting Phase: (a) Candidate nomination and registration process, (b) Voter registration process.

2) Voting Phase: (a) Voter authentication, (b) Vote casting, (c) Vote transmission and confirmation.

3) Post-voting Phase: (a) Counting, (b) Result, (c) Audit administration.

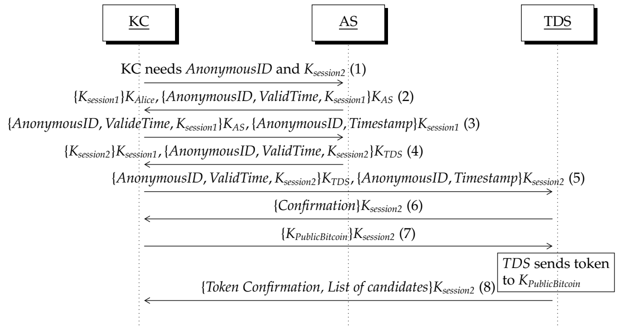
KC
AS
TDS
KC needs *AnonymousID* and $K_{session2}$ (1)
$\{K_{session1}\}K_{Alice}, \{AnonymousID, ValidTime, K_{session1}\}K_{AS}$ (2)
$\{AnonymousID, ValideTime, K_{session1}\}K_{AS}, \{AnonymousID, Timestamp\}K_{session1}$ (3)
$\{K_{session2}\}K_{session1}, \{AnonymousID, ValidTime, K_{session2}\}K_{TDS}$ (4)
$\{AnonymousID, ValidTime, K_{session2}\}K_{TDS}, \{AnonymousID, Timestamp\}K_{session2}$ (5)
$\{Confirmation\}K_{session2}$ (6)
$\{K_{PublicBitcoin}\}K_{session2}$ (7)
*TDS* sends token to $K_{PublicBitcoin}$
$\{Token\ Confirmation,\ List\ of\ candidates\}K_{session2}$ (8)

**Figure 2: The sequence diagram of the Kerberos-based protocol: we have the Kerberos Client (KC), the Authentication Server (AS), and the Token Distribution Server (TDS).**

**Pre-voting Phase.** Step 1(a) is the process of approving nominees as eligible candidates. A candidate in this context can be a named individual or a different entity (e.g., just a statement). The aim is to retrieve a list of eligible candidates owning a couple of asymmetric keys: the public one is associated with the identity of the candidate[9] and it has to be freely available to all the voters, while the related private-key has to be kept secret by each candidate. We let a voter receive the list of verified public-keys at the same time she obtains the voting token, from the same token distribution-service used in phase 1(b). In this setting, we suppose candidates register themselves in person, by showing an ID and communicating their public key directly to the election organiser. To avoid this, a digital registration can be implemented for candidates as well, similar to what proposed for voters in phase 1(b).

Step 1(b) concerns the process of approving voters to successively vote. Due to its nature (e.g., the possibly large number of voters), such a step has to be fully digital. The public key of a registered voter will be charged with an amount of bitcoins, which represents the election token to be spent as a vote.[10] The private key of a voter will be instead used by the voter's wallet to cast a preference in the voting phase, by signing the transaction that transfers her vote. Each voter generates her public/private keys, associated to her wallet.

However, a public key cannot be directly associated with voter's identity, otherwise anonymity of electors would be not guaranteed (see Sec. 5); clearly, anonymity is one of the main properties we need to satisfy (if not "the" property). In order to guarantee it we propose an *Anonymous Kerberos* authentication-protocol [12, 14]. Note that this is only one among different anonymous-authentication approaches that can be exploited to anonymise voters: an alternative can be using a *Blind Signature* scheme [4]. In our implementation (Sec. 4) we have opted for a variant of the Anonymous Kerberos protocol, as explained in the following paragraph.

**Anonymous Kerberos** [12, 14] provides a mechanism for principals to authenticate to a remote service without disclosing their identity. We suppose the sequence is initiated by Alice (one of the voters), who logs into the Kerberos Client (*KC*) with her *username* and *password* (or biometric features). The other two participants involved in the protocol are the *Authentication Server* (*AS*), and the (voting-)*Token Distribution Server* (*TDS*). AS is in charge of authenticating Alice, while secondly TDS transfers (via a Bitcoin transaction) the voting token to Alice by using her public key. In Fig. 2 we show the sequence diagram of the messages exchanged among these three entities. $K_{Alice}$, $K_{AS}$, and $K_{TDS}$ in Fig. 2 are the secret keys of Alice, AS, and TDS respectively: see [12, 14] for a description about how they are created. In order to separate the authentication from a token (i.e., the "ballot"), it is important that AS releases an anonymous credential (i.e., *AnonymousID*) to Alice, used by her to access to TDS together with a session key ($K_{session2}$). In this way, AS will be aware of the identity of Alice without being able to associate it with her public key. On the other hand, TDS will not be aware of the identity of Alice. For the sake of duties separation, AS and TDS need to be implemented by distinct entities: for instance, a delegation of voters may (distributedly) implement AS, and a delegation of candidates may (distributedly) implement TDS. AS and TDS together prevent the same voter to register more than once (and get more than one voting-token): multiple requests with the same *AnonymousID* are denied by TDS.

In the last two messages, 7 and 8 in Fig. 2, first Alice sends (7) her public key ($K_{PublicBitcoin}$), which needs to be "charged" with a voting token. Message 8 confirms that such token has been accredited to the public key (sent by Alice), meaning that TDS has transferred one token to the wallet of Alice, and the related transaction is recorded in the block-chain.

Remark 1 (Multiple Kerberos). *An obvious weakness of the protocol is that, if the two entities AS and TDS collude, then anonymity is immediately broken: AS can match the real identity with the anonymous one, and then match it with the $K_{PublicBitcoin}$ of Alice, with the help of TDS. Even if in our implementation (see Sec. 4) we opt for this simpler scheme, it is indeed possible to complicate it in order to have n entities, and consequently prevent this scenario if $n-1$ (or less) entities collaborate to break anonymity. One solution is to ask Alice (or better, KC) to authenticate $n-1$ times to $n-1$ different ASs: each of them returns a string of data, which Alice concatenates in the right order (AS may have a predefined sequence order), and then hashes it (e.g., with SHA-1). The final string of 160 bits represents the AnonymousID that Alice presents to TDS in order to have its public key charged with one token. TDS is able to discover the real identity of Alice only if it obtains the right sub-string from each AS, and then, by hashing the whole string, TDS matches it with the AnonymousID given by Alice. TDS needs to store the hash of all the possible strings (however, in sequence) of the $n-1$ ASs from the beginning.*
*In order to avoid Alice to authenticate $n-1$ times, the $n-1$ ASs can be concatenated. In this way, Alice authenticates only to $AS_1$, which then authenticates to $AS_2$ and so on, until $AS_{n-1}$ is reached. Each AS encrypts its sub-string by using a public key associated with the real identity of Alice; then it passes the result to the next AS in the chain. Alice will decrypt the final message (obtained from $AS_1$) $n-1$ times before applying the same hash function as before and send the result (i.e., the AnonymousID) to TDS. If only one AS is not maliciously collaborative, AnonymousID cannot be matched with the real identity of Alice.*

**Voting Phase.** After completing the pre-registration, a voter

[9]A Bitcoin (destination) *address* is essentially a hash of the recipient's public-key.

[10]In Sec. 4 we propose two different implementations of a token, based either directly on Bitcoin or the *Open Asset Protocol*, which lead to different election costs.

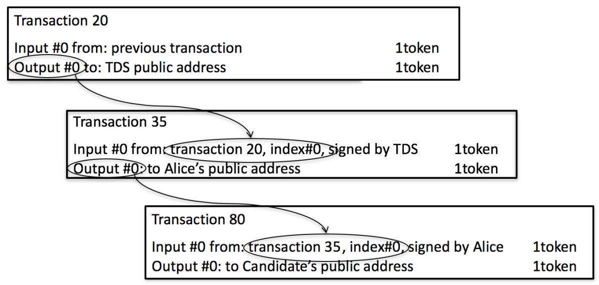
Transaction 20
Input #0 from: previous transaction 1token
Output #0 to: TDS public address 1token
Transaction 35
Input #0 from: transaction 20, index#0, signed by TDS 1token
Output #0: to Alice's public address 1token
Transaction 80
Input #0 from: transaction 35, index#0, signed by Alice 1token
Output #0: to Candidate's public address 1token

**Figure 3: The block-chain excerpt of a valid vote (cast by Alice to Candidate).**

owns a voting-token in her wallet and she is ready to cast her preference. She transfers such token to the Bitcoin address of the candidate she likes. The sub-phase of authentication (2(a)) is performed by signing the transaction with the private key corresponding to the $K_{PublicBitcoin}$ charged in the pre-voting phase. Casting (2(b)) is performed by preparing a payment towards the public key (i.e., address) of the chosen candidate. Transmission (2(c)) corresponds to effectively executing the transaction, while confirmation (still (2(c)) can be done by simply self-checking if a transition is in the block-chain (after that transaction has been mined).

Authentication, casting, and transmission are achieved by operating on the voter's wallet. When Alice wants to send bitcoins (2(b) casting) to Candidate, she uses her private key to sign a message with the input (the source transaction from TDS), an amount (the token), and an output (Candidate's address). The three-transaction chain is represented in Fig. 3. New transactions (votes) are broadcast to the Bitcoin network (2(c) transmission). Miners confirm all the transactions related to votes. All the transactions (i.e., all the votes) in a block are included in the block-chain (2(c) confirmation).

**Post-voting Phase** This phase mainly covers tokens counting and result reporting. *Counting* (sub-phase 3(a)) is indeed the most interesting step. The possibility of recounting needs be considered as well, since it is one of the required properties (see Sec. 5): results need to be confirmed if requested.

Remark 2 (Token). *So far we have referred to the amount of bitcoins transferred in the pre-voting and voting phases with the generic word "token" (i.e., a vote). In the simple case, a token corresponds to the smallest quantity possible of bitcoins that can be transferred in a transaction (one* satoshi, *i.e.,* $10^{-8}$฿*), plus a* transaction fee*, which represents a reward for the miner that adds the new block containing it to the block-chain.*[11] *An alternative is represented by the use of "digital asset coins" (or simply asset coins), that is by using the* Open Asset Protocol[12] (OAP)*. In this way, it is possible to create "assets" on top of Bitcoin, in order to unambiguously mark money, or, here, to uniquely identify voting-tokens. In this case, the token amounts to a different quantity* ($6\times10^{-6}$฿)*, while the fee is the same* ($10^{-4}$฿)*. We have implemented both the solutions, even if the implementation summarised in Sec. 4 describes the OAP alternative (more information and motivations behind using OAP can be found in Sec. 4).*

Votes are counted by summing the tokens obtained by each candidate in the block-chain. As advanced in Rmk. 2, this can be implemented in two different ways: *i)* a token is a satoshi, or *ii)* a token is a digital asset coin (in case of using OAP). While the pre-voting and voting phases are marginally affected by choosing either *i* or *ii*, counting is more involved.

In *i*, to be valid a transaction it needs to both originate from an authorised voter, and end in the address of a registered candidate. In turn, a voter is authorised if she has previously received a token in a transaction from the public key of TDS (see the pre-voting phase). Therefore, for each token in the block-chain, the counting process needs to check if the sub-chain of transactions is identical to the last two transactions in Fig. 3. By remembering all the source addresses of confirmed votes, we enforce that only one vote per authorised voter is counted. Since the block-chain can be accessed sequentially, this kind of search requires $n + 1$ scans of the block-chain (where $n$ is the number of voters): one to count all the votes, while the other $n$ scans to be sure that each vote has been cast by an authorised voter. If we suppose that even the entity that is performing the count is aware of the list of authorised voters (which in principle is known by TDS only), than the block-chain can be accessed only once.

The main reason behind using asset coins, i.e., *ii*, is that OAP is designed to attach metadata to a transaction, and consequently expand Bitcoin functionalities. The token obtained in the pre-voting phase can be then unambiguously marked with the attached metadata signed by an administrator, that is the electoral commission. In this way, bitcoins really become votes, and it is possible to count single votes instead of authorised voters, as in *i* instead. Therefore, there is no need to check if a voter is authorised or not and the count becomes easier: for instance, there is no need to keep track of voters' addresses to check they have voted only once. Without this check however, it is possible for a voter to send her asset coin to a different (authorised or unauthorised) voter, who can cast such a vote without being directly detectable. To avoid such misbehaviour, a possible solution is to use a *permissioned block-chain* [15], where access permissions are more tightly controlled, with rights to write (or even read) the block-chain restricted to a few users. This prevents transactions of among voters, with the purpose to allow transactions only between an authorised voter and a candidate.

To count votes, not all the block-chain needs to be scanned: each block contains a Unix-time timestamp, so that it is possible to count as valid all the transactions in the blocks with a timestamp in the interval *Elections opening* ≤ *timestamp* ≤ *Election closing*. However, the mining process is not immediate: the interval between one block and another has an average of 10 minutes, but not every block interval is exactly 10 minutes, and it follows a Poisson distribution. At the time of writing, in a 10 minute interval the probability of a block being mined is about 63%; in 30 minutes the probability raises to 95%. For this reason and because *Election closing* is not synchronised with the mining process, we are sure to have counted all the votes arrived before the closing if we consider up to two blocks with *timestamp* ≥ *Election closing* in the block-chain.

[11]Fees are required for small transactions, in order for them to be processed without any delay. For transferring one satoshi, a fee of $10^{-4}$฿ is considered as enough to avoid any delay.
[12]http://tinyurl.com/jttcp8y

In addition, the last block (and all the ones before) of the election also needs to be "confirmed", which means not orphaned due to a reorganisation of block-chain due to a fork (see Sec. 2): the Bitcoin community has adopted 6 blocks as a standard confirmation period. Hence, once the block-chain stores 6+2= 8 blocks with *timestamp* ≥ *Election closing*, counting can be launched. The risk of losing a transaction to a reorganisation is low (1-2%), and even then it will probably be re-included after the reorganisation occurs. However, it is better to wait for stable block-chain count only really confirmed transactions: note that counting votes can start as soon as there is a confirmed block with *timestamp* ≥ *Election opening*. On the average, 80 minutes (10min./block) after closing the election is necessary to start the counting.

## 4. IMPLEMENTATION AND COSTS

We have implemented a Web-interface to let a user vote without having a wallet.[13] This online application implements all the three phases described in Sec. 3: the main difference is that there is only one wallet shared by all the (authorised) voters; nevertheless, each voter has different private and public keys (and address) linked to the same wallet. With respect to the other phases, we have implemented an anonymous Kerberos service as described in the pre-voting phase, and the counting as described in the post-voting phase (see Sec. 2). A voter is authorised through her social security number. Indeed we can develop a stand-alone voting application as well, where each voter has her own wallet (see future work in Sec. 6).

For such an implementation we used the following main technologies: *i)* a Web Server Apache, *ii)* a Mysql DBMS, *iii)* Perl CGI, and *v)* a digital-asset wallet compliant with OAP: in this case we use *CoinPrism*[14]. However, the same implementation can work with other OAP-compliant wallets, e.g., *CoinSpark*[15], or *SparkBit*[16]. In Fig. 4 we show the steps of the pre-voting phase in our application: steps 5-6 register a voter, while 8-9 transfer a token to her by using CoinPrism.

CoinPrism API describes a class of methods for representing and managing digital asset coins (called also "coloured coins") on top of the Bitcoin block-chain. While originally designed to be a currency, Bitcoin's scripting language allows to store small amounts of metadata on the block-chain, which can be used to represent asset manipulation instructions. The transaction outputs using OAP to store an asset have two new characteristics: The first one is an asset *ID*, which is a 160 bits hash, used to uniquely identify the asset stored on the output. The second one is an asset *quantity*, which is an unsigned integer representing how many units of that asset are stored on the output. Transactions relevant to OAP must have a special output called the marker output. This allows clients to recognise such transactions. OAP transactions can be used to issue new assets, or transfer ownership of assets.

**Costs.** To compute the cost of the election (in terms of bitcoins) by using OAP, we provide the following three calculations (where $n$ is the number of voters):

[13]The application can be tested at http://evote.dmi.unipg.it/cgi-bin/evote/index.pl.
[14]https://www.coinprism.info.
[15]http://coinspark.org.
[16]https://coinspark.org/sparkbit-wallet/.

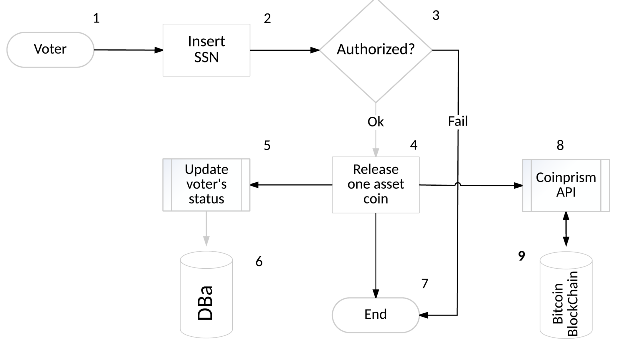

**Figure 4: The architecture of the pre-voting phase.**

a) $\alpha = 10^{-4} + (6 \times 10^{-6})$฿: the cost to issue a new asset with CoinPrism. The cost of issuing or transferring an asset is completely independent from the quantity issued or transferred.

b) $\beta = 2\alpha$, the cost to transfer the asset coin from TDS to the voter, and from a voter to a candidate.

c) $T_{OAP} = \alpha + (\beta \times n) = \alpha \times (2n+1)$, the total cost of the election.

Otherwise, without OAP, by voting with satoshis the cost becomes $T_{noOAP} = (2 \times 10^{-4}$฿ $+ 10^{-8}$฿$) \times n$, which is equal to the cost of a satoshi (i.e., $10^{-8}$฿, the token), plus two times the cost of the fee (i.e., $10^{-4}$฿, the cost to transfer the vote from TDS to a voter and the cost to vote), multiplied by $n$.

To exemplify the cost of an election, we consider as the current price 1฿= 547€, and 1000 voters. In this case, by using the presented platform the total cost (for voting only) would have been $T_{OAP} = 116$ €, and $T_{noOAP} = 109$ € circa. Therefore, the election cost with asset coins (using OAP and CoinPrism) is only marginally more expensive than simply voting with satoshis, and the benefits of having a clearly-marked token to represent a vote can improve the post-voting phase (see Sec. 3). Note that the price is determined by the fee, more than by the token (either a satoshi or an asset coin). Hence, while the same token can be reused for successive elections (the tokens are transferred to known candidates), the fee to transfer them has to be paid again at each election (two times: from a candidate to TDS, and from TDS to each authorised voter). To considerably cut costs, the fee ($10^{-4}$฿) can be removed if waiting for a few days from the time of the election end is negligible: small transactions without any fee have a low priority to be mined. Hence, with 11 €cents per voter it is possible to organise an election with 1000 voters.

## 5. EXPECTED PROPERTIES

We report all the classical properties of e-voting systems [6, 8], and some comments about whether they are satisfied or not satisfied by our E2E voting-system.

**Eligibility and Authentication:** *Only authorised voters are able to vote*; this is accomplished by the pre-voting phase (see Sec. 3). **Verifiability and Auditability:** *It is possible to verify that all the votes have been correctly accounted for in the final tally,* *and there are reliable and demonstrably authentic election records*. The block-chain implements such a public election record, which is public: to modify a block is computationally hard. **Uniqueness:** *No voter is able to vote more than once*. Double-voting is prevented by the fact double-spending of tokens is impossible in Bitcoin (see also the post-voting phase in Sec. 3). **Accuracy:** *Election systems should record the votes correctly, with an extremely small error-tolerance*. The protocol reliability resistance is due to the presence of reliable miners, stimulated by gaining bitcoins as reward. **Integrity:** *Votes should not be able to be modified, forged, or deleted without detection*. When a transaction is confirmed in the block-chain, votes cannot be deleted or modified. If a vote is modified on the client-side, the voter can detect it once the corresponding transaction is in the block-chain. **Verifiability and Auditability:** *It is possible to verify that all votes have been correctly accounted for in the final election tally*. Once again, the block-chain permits this (check the post-voting phase in Sec. 3). **Vote anonymity:** *Neither election authorities nor anyone should be able to determine how any individual voted*. Public-keys of voters cannot be associated with their identity (see the pre-voting phase in Sec. 3). **Counting and Recounting:** *Voting system must provide easy functions for counting and recounting, in case of any question about the final voting result*. Each valid transaction is permanently stored in the block-chain.

Of course, all the properties that imply some kind of privacy on the block-chain to be satisfied are intrinsically problematic: the block-chain is intentionally public by implementation. The unsatisfied properties are: **Uncoercibility and Receipt-freeness.** *Voters should not be able to prove how they voted*. Unfortunately, in our case a voter can prove that she is the source of a transaction registered in the block-chain. Therefore, we can think of using this e-voting system in case the risk of coercibility is low. To prevent this, it is possible to adopt permissioned block-chains [15], where the right to read the block-chain can be granted only to some users. For instance, each voter can read only the block where her transaction is registered in order to still maintain the verifiability property; some official entities can be instead allowed to read the whole block-chain with the purpose to count votes. In this way, unless the coercer is on-site during the voting process, the elector can vote for other candidates. **Data confidentiality and Neutrality.** *Votes must be protected from external reading during the voting process*. Once a vote has been broadcast to the peer-to-peer network, it is not confidential anymore, and can influence successive voters, who can freely read the block-chain and know candidates' addresses. Permissioned block-chains can be used also in this case, in order to prevent other users to scan the whole block-chain and mine all the votes for each candidate.

## 6. CONCLUSION

We have presented an E2E verifiable e-voting system based on Bitcoin. A transaction between a voter and her candidate represents a vote, which is broadcast to the Bitcoin network and verified by miners. We describe a solution that is fully compliant with the current Bitcoin network. Moreover, we also suggest some possible modifications that can improve the performance in counting votes, by using OAP and a permissioned block-chain.

In the future we plan to better investigate all the currently not-satisfied properties presented in Sec. 5: for instance, trying to implement a small Bitcoin-like network with a permissioned block-chain. We believe the further study on permissioned block-chains has strong motivations in such voting applications. In addition, we will release a client-application for voting, with an embedded OAP-compliant wallet.

## 7. REFERENCES

[1] M. Andrychowicz, S. Dziembowski, D. Malinowski, and L. Mazurek. Secure multiparty computations on bitcoin. *Commun. ACM*, 59(4):76–84, 2016.

[2] A. Back. Hashcash - a denial of service counter-measure. http://www.hashcash.org/papers/hashcash.pdf, 2002. [Online; accessed 21-July-2016].

[3] J. Borras and D. Webber. Election Markup Language (EML) Specification Version 7.0. http://docs.oasis-open.org/election/eml/v7.0/cs01/eml-v7.0-cs01.html, 2011. [OASIS, online; accessed 23-Aug-2016].

[4] D. Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology: Proceedings of CRYPTO '82, Santa Barbara, California, USA, August 23-25, 1982.*, pages 199–203. Plenum Press, New York, 1982.

[5] D. Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security & Privacy*, 2(1):38–47, 2004.

[6] L. Fouard, M. Duclos, and P. Lafourcade. Survey on electronic voting schemes. http://www-verimag.imag.fr/~duclos/paper/e-vote.pdf, 2007. [Verimag technical report, online; accessed 21-Sept-2016].

[7] D. A. Gritzalis. *Secure electronic voting*, volume 7. Springer Science & Business Media, 2012.

[8] C. Mote Jr. Report of the national workshop on internet voting: issues and research agenda. In *Proceedings of the 2000 annual national conference on Digital government research*, pages 1–59. Digital Government Society of North America, 2000.

[9] S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. http://www.hashcash.org/papers/hashcash.pdf, 2008. [Online; accessed 21-July-2016].

[10] K. Okupski. Bitcoin protocol specification. http://www.enetium.com/resources/Bitcoin.pdf, 2014. [accessed 19-Sept-2015].

[11] K. Sampigethaya and R. Poovendran. A framework and taxonomy for comparison of electronic voting schemes. *Comput. Secur.*, 25(2):137–153, Mar. 2006.

[12] J. G. Steiner, B. C. Neuman, and J. I. Schiller. Kerberos: An authentication service for open network systems. In *Proceedings of the USENIX Winter Conference.*, pages 191–202. USENIX Association, 1988.

[13] Z. Zhao and T. H. Chan. How to vote privately using bitcoin. In *Information and Communications Security ICICS*, volume 9543 of *LNCS*, pages 82–96. Springer, 2015.

[14] L. Zhu, P. Leach, and S. Hartman. Anonymity Support for Kerberos. RFC 6112 (Proposed Standard), Apr. 2011.

[15] G. Zyskind, O. Nathan, and A. Pentland. Enigma: Decentralized computation platform with guaranteed privacy. *arXiv preprint arXiv:1506.03471*, 2015.