IPSJ Transactions on Mathematical Modeling and Its Applications Vol.10 No.1 14–22 (Mar. 2017)

Regular Paper

E-voting System Based on the Bitcoin Protocol and Blind Signatures

Jason Paul Cruz[1,a)] Yuichi Kaji[2,b)]

Received: January 12, 2016, Revised: March 23, 2016,
Accepted: December 13, 2016

Abstract: This study proposes an electronic voting (e-voting) system based on the Bitcoin protocol and blind signatures. Various cryptographic schemes have been studied to realize secure and efficient e-voting systems, but these systems are hardly used in practical voting. One of the technical reasons for this unfortunate situation is that many e-voting systems require an anonymous communication channel, which is difficult to implement over the Internet. This study investigates if the Bitcoin protocol, a payment network wherein transactions are managed collectively by a peer-to-peer network, can provide a breakthrough to the practicality issue of e-voting systems. In the proposed system, the Bitcoin protocol is complemented with known protocols, such as the blind signature protocol and digital signature protocol, to realize an e-voting system that is secure, anonymous, and transparent. This study discusses several important properties of e-voting systems, including fairness, eligibility, anonymity, robustness, and verifiability, and shows that the use of the Bitcoin protocol brings favorable features besides the anonymity of the communication.

Keywords: electronic voting, Bitcoin protocol, blind signature, information security

1. Introduction

Electronic voting (e-voting) is a promising platform that aims to provide a secure, convenient, and efficient voting environment over the Internet. However, voting through the Internet introduces several concerns, such as fraud, anonymity, and abuse, among others. Previous research have introduced different e-voting systems and protocols with different encryption schemes of varying complexity [1], [2], [3], [4], [5]. These systems make use of a combination of different protocols, such as blind signatures, threshold blind signatures, discrete logarithmic encryption, untappable channels, and anonymous channels or public bulletin boards, to satisfy the most properties that make e-voting systems secure. Even though these systems are comprehensive theoretically, a complete solution that can be implemented in the practical domain is yet to be found. One of the most critical problems in the practical implementation of e-voting systems is the realization of an anonymous communication channel, which is assumed in many theoretical schemes and is considered to be one of the most important feature that can satisfy a number of the properties of e-voting systems. An e-voting system can be considered secure if it satisfies the following properties:

Completeness: An eligible voter is always accepted by the administrator and all valid votes are counted correctly.

Robustness/Soundness: Dishonest voters and other participants cannot disturb/disrupt an election.

Anonymity/Privacy: All votes must be secret; and neither voting authorities nor anyone else can link a vote to the voter who has cast a vote.

Unreusability: A voter cannot vote more than once.

Fairness: Early results should not be obtained, as they could influence the remaining voters.

Eligibility: Only legitimate voters can vote.

Individual verifiability: A voter can verify that his/her vote was really counted.

Universal verifiability: Anybody can verify that the published outcome really is the sum of all votes.

The current study aims to develop an e-voting protocol that makes use of Bitcoin technology to realize an e-voting system that satisfies the properties described above. Bitcoin is an innovative global currency cryptosystem that continuously increases in value and popularity since its launch in 2008 [6]. As of January 2016, Bitcoin has a market capitalization of approximately 6 billion USD, market price per Bitcoin (BTC) of approximately 400 USD, and on average, 200,000 transactions daily [7]. Bitcoin enables a payment system that is secure and decentralized. It is a peer-to-peer network powered by its users, and all Bitcoin transactions are published publicly. Participants agree on a single record of all transactions, which are grouped into blocks, given timestamps, and then published. The hash of each block contains the hash of the previous block to create a chain. Bitcoin provides many desirable properties, including easy mobile payments, reliability, control of one's money, high availability, fast international payments, zero or low fees, protected identity, and privacy [8].

In the current study, the idea is to replace the anonymous communication, which is assumed and needed in many e-voting protocols, with the propagation of Bitcoin transactions. Bitcoin

[1] Graduate School of Information Science, Nara Institute of Science and Technology, Ikoma, Nara 630–0192, Japan
[2] Information Strategy Office, Nagoya University, Nagoya, Aichi 464–8601, Japan
[a)] jpmcruz@ymail.com
[b)] kaji@icts.nagoya-u.ac.jp

transactions are communicated over a peer-to-peer network, and with appropriate management, Bitcoin users (not the Bitcoin addresses) cannot be identified and linked to transactions they create. This feature contributes to enabling anonymity in voting, which is a practical obstruction in many existing e-voting systems.

Bitcoin's protocol and cryptography make the proposed system suitable for an efficient and practical e-voting system wherein the voters are protected and given control over the important aspects of the voting process to protect their vote, privacy, and anonymity, while minimizing the trust and power of other entities to prevent them from performing malicious actions. For example, the administrator cannot introduce dummy votes and the counter cannot falsify the results because only the voters have access to the Bitcoin addresses that will be used for voting and the transactions are recorded publicly in the blockchain.

In Section 2, we will briefly discuss some protocols used in existing e-voting systems. Then, we will analyze specific systems and provide details on the properties they satisfy and their vulnerabilities. Section 3 discusses the Bitcoin protocol to show the security it provides. Section 4 presents some preliminaries and assumptions in using Bitcoin as an infrastructure. In Section 5, we will present our proposed system. Section 6 provides a security analysis of the proposed system based on the defined properties of e-voting systems. Section 7 provides the conclusion and future work.

The authors would like to note that it is not our intention to propose an e-voting scheme that is superior to all other existing scheme. Through this study, we would like to discuss what kind of problems in e-voting can be solved by Bitcoin, and what kind of problems remain unsolved. The discussion will contribute to clarifying the potential of Bitcoin as a general infrastructure over which a secure application is constructed.

2. E-Voting Systems

In this section, we first discuss some widely recognized e-voting protocols, which are the bases of other more complex secret voting schemes that have been proposed by other researchers. Many e-voting systems are based on the blind signature scheme. In this kind of scheme, voters obtain a token or signature, which is created by an authority "blindly" for a data (the vote) that is known only to the voter. The blind signature scheme is simple to understand and is easily adaptable in complicated schemes. However, a common problem with this kind of scheme is its potential for single points of failure because the authorities involved are given too much power, e.g., authorities can introduce dummy votes for voters who abstained from voting. To address this problem, variations of the blind signature have been created. For example, a threshold blind signature scheme (or simply requiring blind signatures from multiple authorities) reduces the power of authorities by creating replicates of these authorities and keeps the protocol from failing as long as a certain number of the replicates are honest. However, this kind of threshold scheme is vulnerable to the problem of colluding voters. To solve this problem, a public bulletin board, which is accessed through anonymous channels, has been used on top of systems based on threshold blind signature schemes to provide transparency in the voting process.

2.1 Blind Signature Scheme

A blind signature, as presented by Chaum [1], is a cryptography that is a digital version of using carbon paper-lined envelopes. In this analogy, the signature written outside the envelope leaves a carbon copy of the signature on the document inside the envelope, which the signer cannot see. In e-voting systems based on the blind signature protocol, the signer (typically an authority) signs an unknown message (blinded message) for a known requester (typically a voter). This blinding process is necessary because the signer should not know the vote of the voter. In this scheme, a data or message to be signed by a signer (authority) is disguised (randomized) first by a provider (voter) using a blinding function given by:

$$m' = blind_e(m, r),$$

where m' is the blinded message, e is the public key of the signer, m is the message, and r is a random number. In an RSA blind signature scheme, the blinding function is defined as $blind_e(m, r) = mr^e \bmod n$, for example. The provider sends m' to the signer, then the signer signs m' to generate the following:

$$s' = sign_d(m')$$

where s' is the blind signature and d is the private key of the signer. Then, the signer returns s' to the provider. The provider obtains the digital signature s for m using a corresponding unblinding function given by:

$$s = unblind(s', r).$$

In the RSA blind signature, $s' = (mr^e)^d = m^d r \bmod n$, and $unblind(s', r) = r^{-1}s' = m^d \bmod n$.

2.2 Secret Voting for Large Scale Elections

Fujioka et al. [2] proposed a voting scheme for large-scale elections. Their model consists of three entity types: voters V_i ($i = 1, 2, \ldots, n$), an administrator A, and a counter C. In this model, V_i and C are assumed to communicate through an anonymous communication channel, which is a virtually assumed special channel where nobody can specify the sender of transmitted data. The model also requires the bit-commitment, digital signature, and blind signatures schemes. Voters V_i have their own digital signature scheme, A has its own blind signature scheme and is responsible for verifying the eligibility of voters, and C only collects the ballots and publishes the results. An informal description of their scheme is as follows:

In the Preparation stage, voter V_i selects a vote v_i and completes the commitment $x_i = enc(v_i, k)$ using the bit-commitment scheme, where k is a randomly chosen key. Then, V_i computes the blinded message $x_i' = blind_e(x_i, r)$, where e is the public key of A and r is a random blinding factor. Then, V_i signs x_i' to generate $s_i = sign_i(x_i')$, where $sign_i(x_i')$ is V_i's signature scheme (hence $sign_i$ is the signature of V_i itself for the blinded messages), and sends this s_i, together with his/her identification and x_i', to A.

In the Administration stage, A verifies the eligibility of V_i. If V_i is eligible to vote, then A signs x_i' to generate a digital signature $d_i = sign_A(x_i')$ for x_i, where $sign_A(x_i')$ is A's signature

scheme. Then, A sends d_i back to V_i. At the end of this stage, A publishes a list that contains the identities, blinded message, and signed blinded message (i.e., ID_i, x_i', and d_i) of the voters that received d_i.

In the Voting stage, V_i retrieves the signature $y_i = unblind(d_i, r)$ for the commitment x_i. Voter V_i verifies the correctness of y_i, and if it succeeds, sends x_i and y_i to the counter C through an anonymous channel.

In the Collecting stage, C verifies that y_i is the signature for x_i. If the test succeeds, C publishes a list of (l, x_i, y_i), where l is the entry number of the corresponding x_i and y_i.

In the Opening stage, V_i checks that the number of ballots that C published is equal to the number of voters that A published, and that his/her vote is listed in the list that C published. If the check succeeds, V_i sends the key k (that was used to make the commitment x_i) with the corresponding number l to C through an anonymous channel.

In the Counting stage, C opens the l-th ballot using the corresponding k to retrieve the vote v_i. Finally, C counts the votes and announces the results.

As pointed out in Ref. [9], this scheme has the potential for a single point of failure, wherein the authority can provide votes for the voters who did not cast their votes.

2.3 Blind Multsignatures

Multisignature schemes are used to avoid single points of failure wherein any subgroup of a group of players jointly sign a document to convince a verifier that each member of the subgroup participated in signing [10]. A multisignature scheme can be combined with a blind signature scheme to create a blind multisignature scheme that can be applied to e-voting systems. In this kind of system, a user runs the blind signature protocol with each signer (the signers are considered to be a subgroup of a group of signers) to obtain signatures of a message, each generated using the corresponding signer's private key. An entity is replicated N times and it assumed that at least t replicates are kept from failing even in the most pessimistic scenarios. The number of t should be greater than 1 and less than N. The value of t should be selected in such a way that it is unlikely that $N - t$ or more replicates collude or fail simultaneously (i.e., it is likely that t replicates stay safe and secure).

A general e-voting system based on blind mutisignatures consists of five entity types: voters V_i ($i = 1, 2, \ldots, n$), an administrator A, registration authorities R_j ($j = 1, 2, \ldots, N$), a key authority K, and a counter C. The stages are as follows:

In the Initialization stage, A distributes the set of identities of legitimate voters together with their corresponding public keys to concerned entities, i.e., the registration authorities R_j with $j = 1, 2, \ldots, N$ and key authority K. In this scheme, K generates the private/public key pairs that will be used for the encryption process during the voting process. Note that K can also be replicated to dissolve its power, but for simplicity purposes, only one K is discussed here.

In the Preparation stage, voter V_i selects a vote v_i and completes the commitment $x_i = enc(v_i, k)$, where k is the public key provided by K.

In the Administration/Registration stage, V_i randomly selects t registration authorities. In this explanation, without loss of generality, let $R_1, \ldots, R_t$ be the authorities selected by V_i. Then, V_i computes the blinded message $x_{i,j}' = blind_{e_j}(x_i, r)$ for $j = 1, \ldots, t$, where r is a random blinding factor and e_j the public key of the corresponding R_j. Note that $x_{i,j}'$ is computed separately for different R_j because it is generated with the e_j of the corresponding R_j. Then, V_i requests a signature for each $x_{i,j}'$ from $R_1, \ldots, R_t$. Then, R_j verifies the eligibility of V_i. If V_i is eligible, R_j signs $x_{i,j}'$ to generate $d_{i,j} = sign_{R_j} x_{i,j}'$. Then, R_j sends $d_{i,j}$ back to V_i. Then, V_i retrieves t signatures $y_{i,j} = unblind(d_{i,j}, r)$ for the commitment x_i. If V_i retrieves the required t signatures, then V_i is ready to vote.

In the Voting stage, V_i sends x_i and the t signatures $y_{i,j}$ to C through an anonymous channel, then C verifies that the required number of t signatures has been satisfied.

In the Opening/Counting stage, K opens the collected commitment x_i publicly using the private key. Then, C counts the votes and announces the results.

In this kind of scheme, K holds the private key for the decryption of x_i, and thus, has the power to reveal the votes as soon as it receives x_i; this can violate the fairness property. Moreover, the different R_j are assumed to not collaborate with each other and they communicate with V_i independently. Therefore, colluding eligible voters can introduce extra vote/s using the extra signatures obtained beyond t.

As pointed out in Ref. [9], a colluding group of voters can introduce extra votes as follows:

$$extra\ votes = \left\lfloor \frac{N - t}{t} V_{col} \right\rfloor \quad (1)$$

where V_{col} is the number of colluding voters. Thus, given $N = 4$ registration authorities and $t = 3$, three colluding voters, $V_{col} = 3$, can generate 1 extra vote.

2.4 Public Registration Boards and E-Voting

Koenig et al. [9] pointed out that a public registration board is required to avoid the problem of colluding voters in e-voting systems based on threshold blind signatures. In this system, they use a public board as a knowledge base for synchronization among the registration authorities. A public board is an append-only broadcast channel with memory or storage device. Data published on the board can be read but cannot be modified.

In their protocol, the voters need not communicate with the registration authorities directly, and vice versa. Following the stages in the system discussed in Section 2.3, they made changes in the Registration and Voting stages. In the Registration stage, voters broadcast a blinded hash of the commitment anonymously to the public registration board. Then, the R_j check the board entries, get the blinded messages, sign the messages, and then broadcast the corresponding blind signatures back to the board. In the Voting stage, voters send the commitment, the hash of the commitment, and the signatures of the R_j anonymously to the counter C. All other stages remain the same.

This kind of scheme solves the problems of colluding voters and single points of failure. However, by introducing the public registration board, it can be prone to some additional problems,

such as denial of service attack and anonymity issues. A natural improvement to this kind of scheme is to create a collective of public boards or a distributed web bulletin board [11]. This collective is based on N peers of identical public boards, which, ideally, have a synchronized history of the entries. Furthermore, the anonymous nature of the bulletin board is still needed, and this approach does not mitigate the practicality issue of previous schemes.

3. The Bitcoin Protocol

Bitcoin is a collection of cryptographic protocols that allow secure online transactions between users [12], [13]. Typically, users own digital Bitcoin wallets that are used for creating and storing private keys and the corresponding *Bitcoin addresses*. A user can transfer or send BTC to another user by creating a *transaction* with the sender's Bitcoin address/es as input/s and the receiver's Bitcoin address/es as output/s. Transactions are validated by *miners* before they are included in a block and permanently recorded in a global public ledger called the *blockchain*.

3.1 Bitcoin Addresses

A Bitcoin address is 160-bit hash of the public key of an Elliptic Curve Digital Signature Algorithm (ECDSA) keypair. The private key is generated first, and then the corresponding public key is derived from the private key. The public key undergoes several cryptographic processes (SHA-256, RIPEMD-160, and Base58 Encoding) to be converted into a valid Bitcoin address which looks like random-like numbers and letters, such as 19zBWfkNicdLdTTweZe37XRj2aFoYmHEX6. Users can create any number of Bitcoin addresses easily and for free. There are 2^{160} possible Bitcoin addresses that can be created, and thus, a Bitcoin address is considered to be "unique" as it is extremely unlikely for two users to independently generate the same Bitcoin address.

3.2 Transactions, Blocks, and the Blockchain

A transaction is a digitally signed data that is broadcasted to the Bitcoin network and then collected into blocks. The general format of a transaction includes a list of input addresses (addresses from which Bitcoins are transferred; these addresses should be outputs of previous transactions), a list of output addresses (which contains the receiving addresses and the amounts of BTC being transferred), and digital signatures (proof that the sender owns the Bitcoin addresses in the list of input addresses). A transaction also has the OP_RETURN feature, which is a script that allows the sender to store 80 bytes of extra data in a Bitcoin transaction [14]. This feature has been used to realize some functions, such as sending messages and proof of existence or timestamping of documents [15], through transactions that are published in the blockchain.

3.3 Blocks and the Blockchain

Transactions are grouped together in blocks and then included in the blockchain. Blocks are connected and linked, wherein each block contains the hash of the previous block to create a chain that connects the first block (genesis block) to the current block. A block contains a hash of the previous block, a hash of the Merkle root of valid transactions to be included in this block, and a nonce (a unique solution to a difficult mathematical puzzle), as shown in **Fig. 1**.

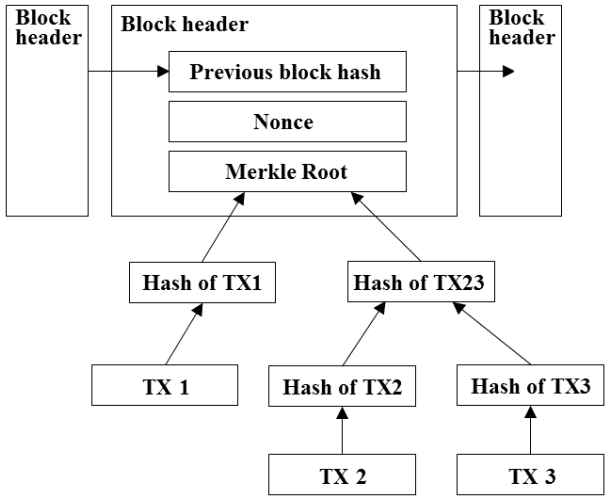

Fig. 1 Simplified representation of the Bitcoin blockchain according to Ref. [5].

The Merkle root is obtained from a Merkle tree, which is a data structure that allows efficient verification of the integrity of data (transactions in our case). A Merkle tree is a binary tree of hashes; leaves are hash values of transactions, and internal nodes are recursively determined as the hash of the pairs of their children. The validity of a transaction can be verified by recursively confirming the hash values along the path from the root to the transaction (leaf) [5]. Transactions are duplicated and broadcasted over the peer-to-peer network of Bitcoin that consists of Bitcoin users. The Bitcoin network itself does not have the mechanism to conceal the IP addresses of the source of a transaction, and, in theory, an IP address can happen to be connected to a Bitcoin address. However, such a risk can be mitigated because full node clients relay transactions as if they are their own transactions, and thus the source of a particular transaction can be difficult to determine unless all communications logs of all nodes are analyzed and traced. Furthermore, third-party services, such as The Onion Router (Tor) [16], can be used to hide the IP address of a computer used in Bitcoin transactions. Online mixing services, such as BitLaundry [17], can also be used, wherein users send and receive Bitcoins to and from such service using independent Bitcoin addresses.

There is a small risk that a Bitcoin user is identified if a Bitcoin address is used in a naive manner. Bitcoin addresses look like random numbers and letters, and the identity of the owner remains unknown unless the same Bitcoin address is used in other transactions where information about the owner is revealed (e.g., buying a product that is delivered physically wherein the name and address of the owner are provided in the product's shipping information). This issue can be avoided if good practices are adopted by Bitcoin users [18]; simply, by using a Bitcoin address only once. A user should not publish his/her Bitcoin addresses in such a way that somebody can connect these addresses with other Bitcoin addresses that are intended for private use. For example, a user is not recommended to move funds from published Bitcoin

addresses to another Bitcoin address that he/she owns. This issue will be revisited later in the discussion of the security.

3.4 Mining and Proof-of-Work

Blocks are added to the blockchain through the process called *mining*, which uses a proof-of-work system wherein miners participating in the Bitcoin network use customized software and hardware to solve mathematical problems. This kind of mathematical problem is inherently difficult to solve and requires a "brute force" solution; that is, miners scan and test for a nonce that gives a correct solution. During mining, the mining hardware of a miner (CPUs, GPUs, FPGAs, and ASICS) runs a cryptographic hashing function composed of two rounds of SHA256 on the block header. The nonce is incremented and hashed together with the block header until a valid solution is found. The solution is governed by a parameter called the difficulty target, which is agreed upon by miners. The difficulty target is expressed as the difficulty on creating the current block compared to generating the first block and is determined as follows:

$$difficulty = \frac{difficulty\_1\_target}{current\_target} \quad (2)$$

As of writing this manuscript, the difficulty is 62,253, 982,449.76 (the probability of each hash to be a valid solution is approximately 6.23×10^{-10}) [7], [19]. The correct nonce should produce a hash value whose numerical interpretation is lower than the difficulty target, or equivalently the hash should start with a certain number of zeroes. When a miner finds the correct nonce, it forwards the solved block to the rest of the miners. After validating the solution for the block, miners move on to determining the correct nonce for the next block. To compensate for increasing hardware speeds, the difficulty target is adjusted every 2,016 blocks so that it takes on average 10 minutes to find a valid nonce. The miner who solves a block is awarded with newly "minted" coins (currently at 25 BTC) and the transaction fees of the transactions included in the solved block. This process of mining guides the issuance of Bitcoins and incentivizes miners to keep mining and approving transactions. The security of Bitcoin relies on this proof-of-work system, which inherently means that a block cannot be modified without redoing the work spent on it, including the work spent on blocks chained after it. Given this design, as long as majority of the overall computing power participating in the Bitcoin network is controlled by honest miners, an attacker will be outpaced by the honest miners, making it almost impossible to modify a published block.

3.5 Voting Schemes Based on Bitcoin

Noizat [20] uploaded a whitepaper that describes a system that leverages the Bitcoin blockchain as a secure transaction database for logging votes and auditing results. He created a demo voting application which can be found in Ref. [23]. His system revolves around a centralized voting application server, which is a point of failure. Consequently, the privacy and anonymity of the voters are violated because the application knows the votes of the voters. Moreover, the application knows the results of the votes as the votes come in, and can know the pattern and behavior of how voters are voting. The application is given almost all the power, such that it handles all the private keys, and it could act maliciously and dishonestly by introducing dummy votes and it can even change the votes.

Zhao and Chan [21] created a fund transfer system that uses the Bitcoin blockchain and realizes a voting-like function. They designed a protocol for the *bitcoin voting problem*, in which voters intend to fund exactly one of two candidates. Some limitations of their system are: the limited number of candidates of only two; the voters need to invest and spend BTC (the amount of BTC needs to be large enough to enforce a penalty system); and the scheme is disrupted if even only one of the voters behaves dishonestly. Their system did not discuss the verification of the eligibility of voters, which is an important property for any voting scheme.

Rockwell [22] created a concept called BitCongress that can create legislation, create elections, and implement legislation changes. It revolves around a front end wallet called AXIOMITY, which holds the digital money and tokens, and is used in conjunction with the Bitcoin, Counterparty, and a Smart Contract blockchains. The AXIOMITY wallet is a point of failure, and the use of three blockchains introduces more probability for the entire system to fail when only one blockchain fails. Moreover, BitCongress violates two important properties of a secure e-voting scheme. First, the votes can automatically be counted as they come in (or are sent to the Bitcoin blockchain), violating the fairness property. Second, the eligibility of the voters cannot be verified because anyone can use the system and automatically get a voting privilege by registering a Bitcoin address in the BitCongress website.

4. Bitcoin as Infrastructure for E-voting

In this section, we investigate the use of the Bitcoin protocol as a substitute to a public board to provide a secure, anonymous, and transparent e-voting scheme. In the proposed system, the voters are given the "power" and control over the sensitive parts of the voting process and the trust to other authorities are minimized. In this way, the voters have complete control over their respective votes, and the authority or counter cannot perform malicious operations because the security of Bitcoin is intact and the transactions are published publicly. In the ideal case, the proposed e-voting system assumes the following:

1. All entities are knowledgeable about Bitcoin, including the protocol and creation of transactions.
2. All entities handle their private keys securely.
3. Voters have initial Bitcoins to spend.

For the first assumption: this is difficult to imagine now as the Bitcoin technology is not easy to understand fully. However, just like any e-voting system, the protocols and methods need to be taught to the voters in an effective manner, similar to teaching Bitcoin. Bitcoin is continuously becoming more widespread and it is currently being used for monetary transactions, proving that the general public can learn the proposed system effectively. Furthermore, this issue can be avoided if an excellent wrapper mechanism that hides the details of Bitcoin from the view of users can be created (e.g., an application or digital wallet that can be used solely for the voting process).

For the second assumption: in a general e-voting system, the voter would need to handle keys (one for encrypting the vote and one used as the blinding factor) privately, securely, and secretly. In the proposed system, the voter would need to handle an additional key, the private key for the Bitcoin address that will be used for voting. The Bitcoin community has provided many methods for securing Bitcoin wallets, including performing backups of wallets (local and online), encrypting the wallet by setting passwords, and creating offline wallets or cold storage [24]. A technology-savvy Bitcoin user can even create multi-signature addresses, e.g., a 2-of-3 multi-signature address, wherein Bitcoins can be transferred using 2 out of 3 private keys used in the creation of the said address. In this aspect, using Bitcoin is advantageous because it provides many security tools that are immediately available.

For the third assumption: this is difficult to implement until Bitcoin becomes a widely accepted currency. In general, users can obtain BTC through Bitcoin ATMs, online and offline exchanges, accepting BTC as payment, by buying from friends, and through mining. In the proposed system, it is ideal that the administrator (or government) provides all necessary Bitcoin expenses, so that the voters do not need to spend any money. However, a voter cannot reveal the Bitcoin address that he/she will use for voting to any other entity without compromising his/her anonymity and privacy. The most viable option is for the administrator to provide Prepaid Bitcoin cards (PBCs) [25] to all eligible voters. PBCs are physical or virtual cards that are pre-loaded with BTC. A PBC contains a public Bitcoin address with a pre-loaded amount of BTC and the corresponding private key, which is covered and can be scratched off, as shown in **Fig. 2**. The PBCs that will be used for the proposed e-voting system can be created by a third-party or by the administrator itself.

Fig. 2 Example of a Prepaid Bitcoin card [25].

5. Proposed E-Voting Protocol

5.1 Overview

The proposed system is a non-conventional e-voting system based on the Bitcoin Protocol and Blind Signature Protocol. The idea is to use the Bitcoin blockchain as an alternative to an anonymous bulletin board or public registration board system, as shown in **Fig. 3**. The proposed system involves three entity types: voters V_i ($i = 1, \ldots, n$), an administrator A, and a counter C. It should be noted that the timing of the Bitcoin transactions is important, i.e., the stages of the proposed system should be followed. If an action is performed outside the timeframe set for a stage, then a vote can become invalid. Nevertheless, such action does not affect the whole system and only affects the entity who performed the action, e.g., only the vote of a voter who breaks the protocol becomes invalid.

5.2 Initialization and Preparation

Administrator A initiates the voting process by publishing empty ballots. Voter V_i selects a vote v_i, completes the ballot, and creates the commitment $x_i = enc(v_i, k)$, where k is a randomly chosen key. Then, V_i generates the blinded message $x_i' = blind_e(x_i, r)$, where r is a random blinding factor and e is the public key of A.

5.3 Administration

This stage is performed in face-to-face communication. Voter V_i requests a signature from A for x_i'. Administrator A checks and verifies if V_i is an eligible voter and has the right to vote and if V_i has not yet requested for a signature. If both conditions are met, A generates the signature $d_i = sign_A(x_i')$, where $sign_A(x_i')$ is A's signature scheme, and then A sends d_i back to V_i. At the end of this stage, when all voters have requested the signature from A, A publishes a list that contains the identities of all the voters who received the signature from A and their corresponding commitment given by $\langle ID_i, x_i' \rangle$.

Voter V_i, who now holds the signature d_i, will retrieve the signature $y_i = unblind(d_i, r)$ for the commitment x_i. Voter V_i verifies if y_i is A's signature for the commitment x_i. If the verification fails, V_i can claim the invalid signature by showing that $\langle x_i, y_i \rangle$ is invalid.

Also during this stage, if PBCs will be issued by A, then V_i is given one of these PBCs (which can be put inside an envelope to ensure that it cannot be traced back to V_i). The voter V_i can check the blockchain to ensure that the Bitcoin address included in the PBC contains coins.

Then, privately, (i.e., at home or some secure place) V_i scratches off the covered private key, allowing him/her to have access to the corresponding Bitcoin address in the PBC. The voter V_i generates a pair of a private key V_i.BPK and a Bitcoin address V_i.BA that will be used for voting. To ensure V_i's privacy and anonymity, V_i transfers the coins from the Bitcoin address in the PBC to V_i.BA. This transaction is performed solely by V_i, and thus, no one, aside from V_i, can link the identity of the owner of V_i.BA to the Bitcoin address included in the PBC. The use of PBCs also introduces an advantage to the security and reliability

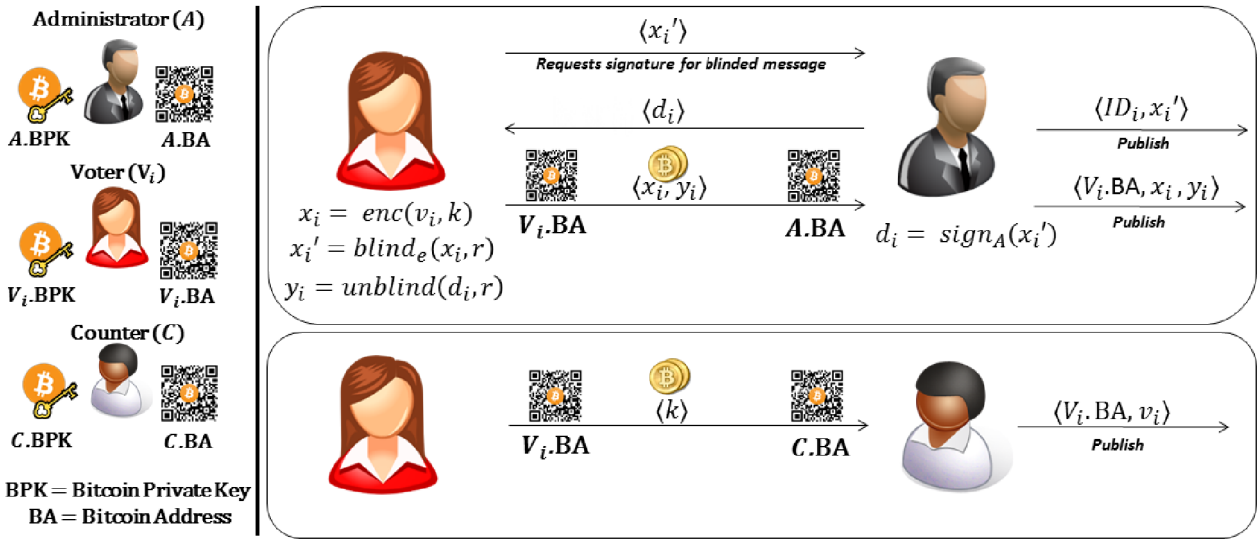

Fig. 3 Overview of the proposed structure.

of the voting process, wherein A publishes all Bitcoin addresses used in the PBCs, and only the Bitcoin addresses that are linked to these PBCs can be used in the voting process.

5.4 Bitcoin Address Registration

This stage is an extension of the Administration stage wherein the eligible voters who received a signature from A will register the Bitcoin Addresses that they will use for anonymous e-voting. From the start of the process, A has generated a pair of a private key A.BPK and a Bitcoin address A.BA. A.BA is published publicly.

The voter V_i creates a simple Bitcoin transaction using V_i.BA as input address and A.BA as output address. In this transaction, V_i includes $\langle x_i, y_i \rangle$ in the OP_RETURN part of the protocol as proof that V_i.BA is owned by a legitimate voter, but the identity of the voter is not revealed. In this transaction, V_i sends an arbitrary amount of BTC to A; currently the minimum amount that can be used for a transaction to be considered valid is 0.00010001 BTC = 0.03 USD (1 satoshi = 0.00000001 BTC plus 0.0001 BTC required transaction fee). Optionally, V_i can include a higher transaction fee or miners' fee if he/she wants the transaction to be prioritized in the current round of solving for the block (but for our purposes, if the time of confirmation is not vital, the minimum transaction fee is sufficient). After confirming the details of the transaction, V_i sends the transaction to the Bitcoin network awaiting for confirmations from miners that it is permanently included in a block in the blockchain.

Once included in the blockchain, certain details will be publicly available, including V_i.BA, A.BA, the amount of BTC transferred, the commitment and signature $\langle x_i, y_i \rangle$ in the OP_RETURN, transaction ID, block number, received time, and the time it was included in the block. Administrator A can verify if the signature y_i of the commitment x_i is valid using its verification key. If the validation is successful, A publishes a list that contains all of the Bitcoin addresses that sent the correct signature y_i of the commitment x_i given by $\langle V_i.\text{BA}, x_i, y_i \rangle$. At the end of this stage, the number of entries in the list that contains $\langle ID_i, x_i' \rangle$ should be equal to the number of entries in the list that contains $\langle V_i.\text{BA}, x_i, y_i \rangle$.

5.5 Voting

Since x_i contains the vote v_i, C can just check and collect the list that contains $\langle V_i.\text{BA}, x_i, y_i \rangle$. Counter C can double-check and verify the content of the list by looking up the transactions in the blockchain using a blockchain browser or a program similar to that.

5.6 Opening and Counting

Voter V_i creates a simple Bitcoin transaction using V_i.BA as input address and C.BA as output address. In this transaction, V_i includes $\langle k \rangle$ in the OP_RETURN. Then, C opens the commitment x_i using the key k to retrieve v_i. Finally, C counts the votes and announces the results by publishing a list $\langle V_i.\text{BA}, v_i \rangle$.

6. System Analysis

6.1 E-voting System Properties

Completeness: If all entities are honest, the results of the voting can be trusted.

Robustness/Soundness: The entities hold their own private keys, thus no other entity can perform a function or transaction on their behalf. Possible cases that can disrupt the voting process are as follows:

During the Initialization and Preparation stage, a voter V_i can keep sending invalid ballots, either with an invalid vote v_i or commitment x_i. However, this can be detected in the Counting stage, and the invalid vote will not be counted. Moreover, V_i can receive only one signature y_i from A for one commitment x_i, and x_i (hence the contents of the ballot) cannot be changed.

During the Counting stage, if V_i sends an illegal key k that cannot open x_i and obtain v_i, the fault can only come from a dishonest V_i because all Bitcoin transactions are published publicly and only V_i has access to the Bitcoin address used for the voting and only V_i possesses k that opens x_i.

After the Bitcoin Address Registration stage, if V_i leaks the pri-

vate key V_i.BPK to others, a corrupt third-party cannot introduce a new v_i because it cannot change x_i. If V_i loses V_i.BPK before sending k to C, then V_i can use another Bitcoin address and state that he/she is the owner of the compromised Bitcoin address by providing the correct k that opens x_i to obtain v_i (This can be safely assumed to be valid because only the eligible voter possesses the correct k and x_i cannot be changed). If V_i loses V_i.BPK after sending k to C, then there is no problem as the purpose of the Bitcoin address was already fulfilled; i.e., x_i has been cast and k that opens x_i to obtain v_i has already been sent.

The Administrator A cannot introduce additional votes by using its own Bitcoin addresses and creating dummy signatures because the entries in the list $\langle V_i.\text{BA}, x_i, y_i\rangle$ will overflow and it will not match the list $\langle ID_i, x_i'\rangle$. A mismatch or overflowing of the lists can only happen because of a corrupt A. Moreover, A cannot dummy vote for a voter that did not cast a vote because it cannot reproduce the voter's commitment x_i and generate x_i', which is published in $\langle ID_i, x_i'\rangle$.

Anonymity/Privacy: The relation between V_i's identity (ID_i) and x_i is hidden by the blind signature scheme. The information that V_i sends through Bitcoin transactions (x_i and k) maintain V_i's anonymity but can be considered valid because of the signature y_i, which can only be obtained by an eligible V_i. The Bitcoin address used by V_i in the voting process, i.e., V_i.BA, cannot be linked to the identity of V_i if proper management is taken and if V_i uses this Bitcoin address solely for the voting process (see also Section 3.3). Moreover, the vote v_i remains secret until the Opening and Counting stage when V_i sends the key k that opens x_i.

Unreusability: It is assumed that the blind signature and cryptographic schemes cannot be broken. Voter V_i can be given only one signature y_i for one commitment x_i. Therefore, V_i can only register one valid V_i.BA in the Bitcoin Address Registration stage. If V_i uses other Bitcoin address/es to send the same pair of x_i and y_i, this redundancy can easily be detected because all Bitcoin transactions are published publicly in the blockchain, and thus, V_i's vote, which is connected to only one Bitcoin address, can only be counted once.

Fairness: The votes are encrypted and disguised using the encryption scheme, and thus, they cannot affect the voting during the Voting stage. Moreover, the Opening and Counting stage is performed after the Voting stage; i.e., V_i sends k that opens x_i to obtain v_i only during the Opening and Counting stage. A voter can intentionally disrupt fairness by sending k during the Voting stage. To avoid this, C.BA can be kept secretly, and then publicly announced only during the Opening and Counting stage.

Eligibility: Assume that the digital signature scheme used by A cannot be broken, and thus, V_i cannot generate a correct signature y_i for x_i on his/her own. The verification of the eligibility of voters is performed in the Administration stage.

Individual verifiability: Given that Bitcoin transactions are published publicly, V_i can easily verify that v_i should be included in the counting by checking the blockchain using a blockchain browser. This also means that a corrupt A cannot exclude a Bitcoin address that sent the pair of $\langle x_i, y_i\rangle$. A corrupt C cannot exclude v_i because k is sent publicly and is easily verifiable. A corrupt C cannot give a false tally of the results as all votes are publicly available.

Universal verifiability: The published outcome cannot be falsified because all votes are public.

6.2 Computational and Transaction Costs

In general, the computational costs are very low and do not require much computing power. The calculations needed for the encryption of the vote, the signature generation of the token from the Administrator, and the blinding and unblinding processes require minimal power and can be performed using current devices. Moreover, the processes involving the Bitcoin protocol, including the creation of the keypair of the private key and the corresponding Bitcoin address and the creation of transactions, are typically performed using digital wallets that can be installed in mobile phones or accessed through the Internet. For the transaction cost, in the proposed system, the voters would need to make at least two transactions, one for the registration of the Bitcoin address that will be used for the voting and another for the sending of the key that will open the commitment. If PBCs will be issued by the Administrator, then the voter would need to create one more transaction to transfer the BTC in the PBC to the Bitcoin address that will be used for voting. As mentioned in the previous section, a Bitcoin transaction only costs 1 satoshi plus the transaction fee, which is approximately 0.03 USD, and thus can be considered quite cheap and affordable. Depending on the protocols set on the voting process, the number of satoshis can increase if multiple transactions are needed to send the required information through the OP_RETURN script, but the transaction fee remains the same.

7. Conclusion and Future Work

This study uses the Bitcoin protocol to realize an e-voting system that is secure, anonymous, and transparent. The Bitcoin protocol is complemented with well-known protocols of existing e-voting systems to provide an alternative for public bulletin boards and avoid the issue of anonymous communication channels that introduces problems in many existing schemes. Compared to other e-voting systems, the proposed system provides power and control to the voters while minimizing the trust on other entities. In the proposed system, PBCs are recommended for use to ensure the privacy and anonymity of the voters. The issuance of PBCs is done physically in face-to-face communication and it can become practically inconvenient if the number of voters becomes large. Thus, future research will focus on the development of digital methods to distribute PBCs securely and conveniently. We will also develop a prototype that can demonstrate possible scalability and to make the proposed system easier to understand and use. We will also investigate the Ethereum protocol [26] if it can be used for the practical implementation of the proposed system. Besides e-voting, it will be interesting to investigate utilizations of the Bitcoin protocol for different security applications.

References

[1] Chaum, D.: Blind Signatures for Untracable Payments, *CRYPTO '82*, pp.199–203 (1982).
[2] Fujioka, A., Okamoto, T. and Ohta, K.: A Practical Secret Vot-

ing Scheme for Large Scale Elections, *AUSCRYPT '92*, pp.244–251 (1992).
[3] Benaloh, J.C. and Yung, M.: Distributing the Power of a Government to Enhance the Privacy of Voters, *5th ACM Symposium on Principles of Distributed Computing*, pp.52–62 (1986).
[4] Okamoto, T.: Receipt-free electronic voting schemes for large scale elections, *Security Protocols*, pp.25–35 (2005).
[5] Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System (2008) (online), available from ⟨https://bitcoin.org/bitcoin.pdf⟩ (accessed 2016-01-27).
[6] Okamoto, T.: An Electronic Voting Scheme, *IFIP '96 Proceedings*, pp.21–30 (1996).
[7] Blockchain Info: Currency Stats (online), available from ⟨https://blockchain.info/stats⟩ (accessed 2016-01-27).
[8] Bitcoin.org: Bitcon for Individuals (online), available from ⟨https://bitcoin.org/en/bitcoin-for-individuals⟩ (accessed 2016-01-27).
[9] Koenig, R., Dubuis, E. and Haenni, R.: Why Public Registration Boards are Required in E-Voting Systems Based on Threshold Blind Signature Protocols, *Electronic Voting*, pp.255–266 (2010).
[10] Boldyreva, A.: Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme, *PKC*, pp.31–46 (2003).
[11] Heather, J. and Lundin, D.: The append-only web bulletin board, *Formal Aspects in Security and Trust*, pp.242–256 (2009).
[12] Bitcoin Wiki: Protocol Specification (online), available from ⟨https://en.bitcoin.it/wiki/Protocol_specification⟩ (accessed 2016-01-27).
[13] Bitcoin.org: Bitcoin Developer Guide (online), available from ⟨https://bitcoin.org/en/developer-guide#block-chain⟩ (accessed 2016-01-27).
[14] BitcoinWiki: Script (online), available from ⟨https://en.bitcoin.it/wiki/Script⟩ (accessed 2016-01-27).
[15] Proof of Existence: Proof of Existence (online), available from ⟨https://proofofexistence.com/⟩ (accessed 2016-01-27).
[16] The Onion Network: Tor (online), available from ⟨https://www.torproject.org/⟩ (accessed 2016-01-27).
[17] BitLaundry: BitLaundry (online), available from ⟨http://bitlaundry.appspot.com/⟩ (accessed 2016-01-27).
[18] Bitcoin.org: Protect your privacy (online), available from ⟨https://bitcoin.org/en/protect-your-privacy⟩ (accessed 2016-01-27).
[19] Blockexplorer (online), available from ⟨https://blockchain.info/q/getdifficulty⟩ (accessed 2016-01-27).
[20] Noizat, P.: Using the Bitcoin Blockchain for secure, independently verifiable, electronic votes, whitepaper (online), available from ⟨https://bitcoin.fr/public/divers/docs/Blockchain_Electronic_Votes_-_White_paper.pdf⟩ (accessed 2016-01-27).
[21] Zhao, Z. and Chan, H.: How to Vote Privately Using Bitcoin, *ICICS 2015 Proceedings*, pp.82–96 (2015).
[22] Rockwell, N: BitCongress, whitepaper (online), available from ⟨http://bitcongress.org/BitCongressWhitepaper.pdf⟩ (accessed 2016-03-14).
[23] Noizat, P.: unchain.voting (online), available from ⟨http://www.unchain.voting/bootstrap/index?l=en&locale=fr⟩ (accessed 2016-01-27).
[24] Bitcoin.org: Securing your wallet (online), available from ⟨https://bitcoin.org/en/secure-your-wallet⟩ (accessed 2016-01-27).
[25] Prepaid Bitcoin (online), available from ⟨https://prepaidbitco.in/⟩ (accessed 2016-01-27).
[26] Ethereum Frontier: Release (online), available from ⟨https://www.ethereum.org/⟩ (accessed 2016-01-27).

Jason Paul Cruz was born in Manila, Philippines, on May 8, 1988. He received his B.S. degree in Electronics and Communications Engineering and M.S. degree in Electronics Engineering from the Ateneo de Manila University, Quezon City, Philippines, in 2009 and 2011, respectively. He is currently a Ph.D. candidate in Graduate School of Information Science, Nara Institute of Science and Technology, Nara, Japan. His current research interests include role-based access control, blockchain technology (Bitcoin and Ethereum), hash functions and algorithms, and Android programming.

Yuichi Kaji was born in Osaka, Japan, on December 23, 1968. He received his B.E., M.E., and Ph.D. degrees in Information and Computer Sciences from Osaka University, Osaka, Japan, in 1991, 1992 and 1994, respectively. In 1994, he joined Graduate School of Information Science, Nara Institute of Science and Technology, Nara, Japan. In 2003 and 2004, he visited the University of California Davis and the University of Hawaii at Manoa as a visiting researcher. He joined Nagoya University in 2016. His current research interests include the theory of error correcting codes, fundamental techniques for information security, and the theory of automata and rewriting systems. He is a member of IPSJ and IEEE.