SCIS 2019 2019 Symposium on
Cryptography and Information Security
Shiga, Japan, Jan. 22 - 25, 2019
The Institute of Electronics,
Information and Communication Engineers

Generic Analysis of E-voting Protocols by Simplified Blockchain

Seunggeun Baek * Nabi Lee * Kwangjo Kim *

Abstract: Upon blockchain technology gains popularity, lots of attempts have been made to adopt this into all ICT systems. One notable example is to apply blockchain to e-voting protocols. However, careful analysis on the requirements and side effects should be scrutinized before the adoption of the new technology. We analyze whether blockchain-based e-voting protocols still satisfy their security requirements, by deriving generic transformations from blockchain-based e-voting protocols to classical e-voting protocols, while preserving the knowledge set for the public. Compared to classical e-voting protocols, we claim that blockchain-based e-voting protocols have *generically weaker* confidentiality and *generically stronger* universal verifiability, judged by the point of our generic analysis.

Keywords: blockchain, e-voting

1 Introduction

Voting is said to be the foundation of democracy to make consensus to solve conflicts efficiently in a group, an organization, and a country, etc. Electronic voting (e-voting) has a long history of research to provide elections with convenience and reliability. To fulfill such demands, there have been several approaches over the cyberspace to apply cryptographic tools or cutting edge technology.

As blockchain technology emerges, voters expect that data integrity in e-voting systems can be kept with 100% correctly by applying blockchain technology, which can be useful to create persistant records that are extremely difficult to tamper with [19]. However, different from cryptographic voting protocols, many blockchain-based voting protocols [2, 8, 9, 10, 12, 15, 18, 21, 22] are designed and implemented, sometimes without proper care of the security requirements. In this paper, we suggest simplified blockchain model for abstraction of blockchain. Through the generic transformations under simplified blockchain model, we demonstrate how the adoption of blockchain affects the e-voting protocols in the context of the requirements, deducing which point the blockchain-based e-voting is strong or not.

Protocol Comparison We compare protocols in the context of a given requirement as follows:

Definition 1. For a requirement r and sets of e-voting protocols X and Y, X has *generically stronger* r compared to Y (or, Y has *generically weaker* r compared to Y) if $\forall y \in Y$, $\exists x \in X$ such that if y satisfies r, x satisfies r.

* Graduate School of Information Security, KAIST. 291, Daehak-ro, Yuseong-gu, Daejeon, South Korea 34141. baek449, butterfly2, kkj @kaist.ac.kr

1.1 Our Contribution

Simplified Blockchain Model We constructed simplified blockchain model to interpret blockchain in the generic domain.

Generic Transformations We derive generic transformations between blockchain-based e-voting protocols and classical e-voting protocols, based on simplified blockchain model.

Requirement Analysis From the generic transformations, we claim that transformed e-voting protocols have *generically weaker* confidentiality and *generically stronger* universal verifiability compared to original protocols.

1.2 Outline of the Paper

In Section 2, we recall previous approaches related to this paper. In Sections 3 and 4, we introduce generic voting phases and the simplified blockchain model, respectively, which is the major tool we use to develop generic arguments. In Sections 5 and 6, we provide generic transformations between blockchain-based e-voting protocols to public bulletin-board-based protocols that preserve confidentiality and universal verifiability requirements of e-voting, respectively. We provide some examples of transformation in Section 7.

2 Previous Approaches

The e-voting protocol is considered to be a set of automatic procedures in which ballots cast by voters are stored and counted in the system. Classically, cryptographic voting is the mainstream of the voting protocol research, which mostly relies on various cryptographic primitives. While cryptographic voting protocols construct provable security to fulfill cryptographic requirements, blockchain-based protocols utilize the structure of blockchain to ensure the integrity of the votes.

2.1 Cryptographic E-Voting Protocols

Chaum et al. [3] proposed Scantegrity voting protocol, which combines end-to-end (E2E) systems' cryptographic ideas with the familiarity of the current paper-based system. The protocol thus provides security guarantees for E2E systems but is unobtrusive to voters.

Civitas [4] is the first e-voting protocol that is coercion-resistant, universally and individually verifiable, and suitable for remote voting. Civitas uses zero-knowledge proof during key generation and decryption, achieving provable security on the random oracle model.

Lee et al. [14] proposed a receipt-free mixnet-based e-voting protocol by using the re-encryption technique and designated verifier re-encryption proof (DVRP).

On the other side, researchers have studied to mitigate instances exploiting vulnerabilities of protocols. Lee and Shahandashti [16] gave a survey of existing attacks against E2E e-voting protocols such as protocols based on Chaumian mix-nets and homomorphic mix-nets [13], and so on. They reconfirmed that integrity and privacy, especially coercion resistance are crucial security requirements of e-voting systems.

2.2 Applications of E-voting

Cortier et al. [5] analyzed a voting system used for electing political representatives and in citizen-driven referenda in the Swiss canton of Neuchatel via ProVerif. The authors showed that the Neuchatel protocol guarantees privacy and ensures cast-as-intended and recorded-as-cast verifiability against dishonest voting servers.

This study is closely related to the analysis of the Norwegian protocol [7], which was used in September 2011 and September 2013 for municipal and county elections in Norway. Norwegian protocol focused on ballot privacy by explaining general lemmas regarding equivalence. Both papers are significant as they proposed a formal analysis of protocols used in real world.

2.3 Blockchain-based Voting Protocols

Zhao and Chan [22] suggested a e-voting protocol in 2015, which is the first approach to combine e-voting with blockchain, especially Bitcoin. As a deposit-based protocol, they introduce a penalty scheme for dishonest behavior of voters without a central authority(CA). In the protocol, a masked vote for each voter is generated via zero-knowledge proof. A transaction for the masked vote is recorded on the blockchain, so that the winner is guaranteed to receive the prize digitally and any nodes deviating from the protocol will lose the deposit.

Lee et al. [15] proposed an blockchain based e-voting protocol working on Bitcoin, which emphasizes the role of a trusted third party between an authenticating CA and a voter. Bistarelli et al. [2] presented an E2E voting-system based on Bitcoin using anonymous Kerberos for identification and authentication. The authors described a solution that is fully covered with the current Bitcoin network using a permissioned blockchain.

McCorry et al. [18] proposed the implementation of a decentralized and self-tallying internet voting protocol, called Open Vote Network (OVN), tested on Ethereum testnet with 40 simulated voters. Compared to previous blockchain e-voting protocols, OVN does not depend on any trusted authority to count the tally and consists of a decentralized two-round protocol designed for supporting small-scale boardroom voting.

Hanifatunnisa and Rahardjo [9] proposed database recording of voting results using blockchain. Recording hash values of the voting results makes the voting system more secure and reliable.

Hjalmarsson et al. [12] evaluated some of the popular blockchain frameworks and proposed an e-voting system using smart contracts on private blockchain. Exonum, Quorum and Geth are smart contracts considered for implementing and deploying the system.

Hardwick et al. [10] proposed an e-voting scheme based on blockchain technology to store cast ballots so that the blockchain acts as a transparent ballot box. Also, every voter is responsible for filtering fraudulent votes because they can audit and verify cast ballots.

Dagher et al. [8] proposed a blockchain-based voting system named BroncoVote. BroncoVote implements a university-scaled voting framework using Ethereum based on smart contract and cryptographic techniques including homomorphic encryption.

Yu et al. [21] proposed a platform-independent voting system with smart contracts, making the protocol independent on a CA for tallying and result announcing. The authors employ Paillier homomorphic encryption [20] to count ballots without leaking candidate information. The protocol is simulated with 1 million voters to evaluate scalability.

3 Voting Phases

A voting protocol involves many parties listed here.

- CA: Central authority
- RG: Registrars
- P: A set of entities (may include non-eligible entities for voting)
- V_E: A set of eligible voters
- V_{Reg}: A set of registered voters
- B: A storage of cast ballots, i.e. a ballot box
- T: A set of tallyers

Here, we define some generic voting phases using the popular conventions as follows:

3.1 Registration Phase

The registration phase requires a CA (the holder of V_E), the registrars, and the target person who wants to participate in the vote.

Algorithm 1 Generic Registration Phase

Input: CA, RG, $p \in P$
Output: t_p or $\perp$
1: $x \leftarrow data(p)$
2: **if** $(eligibilityCheck(CA, p, V_E) = \perp)$ **then**
3: **return** $\perp$
4: **end if**
5: **if** $(convince(CA, RG, p) = \perp)$ **then**
6: **return** $\perp$
7: **end if**
8: $V_{Reg} \leftarrow V_{Reg} \cup \{p\}$
9: $t_p, x \leftarrow makeToken(RG, p, x)$
10: **return** t_p, x

Line 3 in **Algorithm 1** indicates the situation that the CA rejects registration as p is not eligible. In this case, p will formally claim to the CA if $p \in V_E$ actually and p wants to vote. Meanwhile, in protocols which $CA = RG$, $convince(CA, CA, p)$ always return $\top$.

If the convincing process is done without any problem, p is registered to V_{Reg} and an eligibility token t_p is issued for p. For privacy, many protocols import blind signature schemes in $data$ and $makeToken$.

3.2 Vote Casting Phase

The vote casting phase requires registered voters p and a storage of submitted votes B. Voters make ballots containing their selections and eligibility tokens before casting ballots to B.

Algorithm 2 Generic Vote Casting Phase

Input: B, $p \in V_{Reg}$, $selection_p$
Output: ϕ
1: $b \leftarrow makeBallot_p(selection_p)$
2: $B \leftarrow B \cup \{b\}$
3: **return**

The integrity of B should be ensured during this process, so that anyone should not be able to modify the contents in B. The protocol may use a public B; B acts as a public bulletin board. Later we will provide conversions from public blockchain-based voting protocols to bulletin boards.

3.3 Tallying Phase

The tallying phase requires tallyers T as well as the cast ballots B and the registered voters list V_{Reg}.

Algorithm 3 Generic Tallying Phase

Input: T, B, V_{Reg}
Output: $result$
1: $triggerTally_P(T, B)$
2: $B' \leftarrow prepareTally(T, B)$
3: $a \leftarrow []$
4: **for** $b \in B'$ **do**
5: $a \leftarrow count(a, b)$
6: **end for**
7: **return** $announce(T, P, a)$

Some blockchain-based protocols, especially protocols using self-tallying smart contracts, are designed so that everyone can tally the ballots with transparency. This means T is equal to P and $announce(T, P, a)$ does nothing as everyone already knows the result a.

4 Simplified Blockchain Model

Blockchain can be modelled as an append-only distributed ledger of which integrity is assured by consensus algorithms among all participating nodes. In the low-level of blockchain, node management determines network and mining aspects of participating nodes. Consensus algorithms such as Proof of Work(PoW), etc. ensure that each participating node has a common shared ledger.

These low-level aspects of blockchain are diverse according to the concrete implementations, making no significant functional abstractions for blockchain. Therefore, we utilize the following simplified blockchain model as a high-level generic abstraction of blockchain.

4.1 Simplified Blockchain as an Append-only Storage

Blockchain commonly has two interactions with users. A user may read the data recorded in the blockchain, or append new data into the previous block of blockchain.

Definition 2. A simplified blockchain BC is defined by

$$BC := \begin{cases} R_{BC} \subset R \\ read_P : T \to R \cup \{\perp\} \\ append_P : R \to T \cup \{\perp\}, \end{cases} \tag{1}$$

where R is a set of records, P is a set of parties, T is a set of tags, R_{BC} indicates records possessed by the simplified blockchain, $read$ and $append$ are the read and append oracles.

Tags provide metadata of records; for example, the block number is a kind of tags for records inside the blocks. The read oracle $read$ returns corresponding record of the given tag, while the append oracle $append$ writes new record in the simplified blockchain.

In the case of private blockchain, $read$ and $append$ oracles first check that the party triggered the oracle has the suitable privilege, and return the failure symbol $\perp$ to deny the action. For public blockchain, $read$ returns $\perp$ only when the tag is invalid.

Note that the simplified blockchain structure is similar to the structure of bulletin board, which is defined as below:

Definition 3. A bulletin board BB is defined by

$$BB \leftarrow \begin{cases} R_{BB} \subset R \\ read_P : T \to R \cup \{\perp\} \\ write_P : R \to T \cup \{\perp\} \\ delete_P : T \to \{\top, \perp\}, \end{cases} \tag{2}$$

while the rewriting oracle $rewrite_P : T, R \to T \cup \{\perp\}$ can be implemented by the concatenation of *delete* and *write*.

When a bulletin board is public so that any entities can read, the *delete* oracle does not have to exist. The reason is, assuming no one forgets the knowledge on the records he or she already possessed, the *delete* oracle is only meaningful if an entity newly acquires read permission on the bulletin board. Now the setting is identical with the high-level view of simplified blockchain; this is why we can replace a bulletin board readable by the public managed by a honest CA into a public simplified blockchain, and vice versa. Such notions of public bulltin board have been used for cryptographic e-voting constructions, like the notion proposed by Hauser and Haenni [11] using *Get* and *Post* instead of *read* and *write*.

Depending on the writing privilege, a public bulletin board writeable by the public or some specific entities can be replaced by a permissionless blockchain or a permissioned blockchain, respectively.

4.2 Simplified Blockchain as a Smart Contract Platform

Blockchain is often used as a platform uploading and executing smart contracts. For the sake of simplicity, one may distinguish smart contract instructions from the other records written in the blockchain.

Definition 4. A simplified blockchain with smart contract BC' is defined by

$$BC' := \begin{cases} R_{BC'} \subset R \\ contract \subset R \\ read_P : T \to R \cup \{\perp\} \\ append_P : R \to T \cup \{\perp\}, \end{cases} \tag{3}$$

where *contract* is a sequence of instructions.

Common smart contract based e-voting protocols include self-tallying protocols and deposit-based protocols. Self-tallying protocols perform tallies on the public space without dedicated tallyers. Deposit-based protocols require voters cryptocurrency deposits in order to ensure honest behavior.

4.3 Record-independence

For the protocols using the capabilities of blockchain structure, we define record-independence of a blockchain-based protocol V in simplified blockchain model.

Definition 5. A protocol V over blockchain BC with records R_{init} at the start of the protocol is record-independent if $\forall p \in P, t \in tags(R_{init})$, V does not execute $read(p, t)$.

Informally, a protocol V is record-independent if all aspects in V do not have dependency on the state of blockchain written prior to the initiation of the protocol. Deposit-based e-voting protocols are not record-independent since they should query cryptocurrency balances of the voters, executing *read* on past transactions of the voters.

5 Confidentiality Requirements

Confidentiality requirements are constraints on a set of knowledge of each participating parties of the voting protocol, for the sake of our simplicity in the paper. Confidentiality requirements include, but are not limited to, following specific requirements.

Privacy An adversary should not be able to guess which entity the voters voted, in more advantage than guessing with the final results only. To break the requirement, the adversary uses the public records and the private knowledge which the adversary holds.

Coercion Resistance Voters should not be able to prove which votes they cast to any other entity. This is crucial to prevent vote coercion and vote buying. Receipt-freeness is one of the properties required to achieve coercion resistance. To break the requirement, the adversary uses public records, the private knowledge the adversary holds, and the knowledge of the victim voter.

Intuitively, operations on records either on a public bulletin board or public blockchain do not affect the private knowledge of each party. Knowledge of public parties on the records seems identical regardless of whether bulletin boards or blockchains are used. Therefore, we can predict no confidentiality difference in both protocols.

5.1 Record-dependent Protocols

We first assert that blockchain-based protocols which are not record-independent might be vulnerable themselves against the confidentiality requirements. This is because the dependencies to the past records of blockchain becomes side knowledge of an adversary which may infringe the confidentiality of voting.

For example, consider a deposit-based e-voting protocol that requires some cryptocurrency in the balance for every voter. Voters would already have done some transactions to own cryptocurrency to participate in the vote, and these records can be accessed by the public. Any entities who had transactions with victim voters before, such as the exchange markets, may become adversaries who relate the voting account to the real identity of the victim. Recent technologies such as transaction graph analysis can bypass defensive mitigations such as cryptocurrency laundry to some extent.

5.2 Generic Transformations

To claim that blockchain-based e-voting protocols have no confidentiality advantage over normal e-voting protocols, we suggest generic compilers that transform record-independent blockchain-based voting protocols into classical e-voting protocols, under our simplified blockchain model.

Simplified Blockchain without Smart Contract

Algorithm 4 describes how to make a public bulletin board based e-voting protocol using a simplified blockchain-based protocol invoking *read* and *append* oracles.

Algorithm 4 Conversion from Simplified Blockchain

Input: $V(read_p(t), append_p(r))$
Output: A public bulletin board based protocol

1: $BB \leftarrow \begin{cases} R_{BB} = \phi \\ read'_P : T \rightarrow R \cup \{\perp\} \\ write_P : R \rightarrow T \cup \{\perp\} \\ delete_P : T \rightarrow \{\top, \perp\} \end{cases}$
2: **return** $V(read'_p(t), write_p(r))$

This conversion is simple; one may establish a public bulletin board and replace *read* and *append* oracle invocations into the *read* and *write* functions, respectively.

Simplified Blockchain with Smart Contract

Definition 6. An emulator $E_{BB}(c)$ of a smart contract c on a bulletin board BB is a virtual entity which executes instructions corresponding to each smart contract function upon request.

The emulators are executed by the manager of the bulletin board, typically the CA. For each instruction i in a smart contract c, $E_{BB}(c)$ on a public bulletin board BB may convert the instructions in Table 1.

Algorithm 5 describes how to transfer each function from the layer of smart contract into the layer of the CA's protocol emulations. The resulting bulletin board should store the initial state of the contract, and each element of the smart contract becomes a corresponding emulator call in the protocol emulations.

In order to perform conversion, the emulator should be able to simulate most of the operations of smart contracts on the instruction level. The only exception is to allow the access to the environment dependent on the blockchain state before the protocol starts, as the execution of the protocol should be independent to the previous state of the blockchain. For instance, cryptocurrency balance is one such record; balance calculation and value transfer are the affected operations. From the record-independence of a protocol, we can safely predict that such operations will not occur in smart contracts to be successfully emulated.

Through these transformations, one can generically derive classical e-voting protocols using record-independent protocols in simplified blockchain model. For protocols that are record-dependent, we already have pessimistic predictions for the confidentiality of such protocols. Therefore, we claim that "*confidentiality of blockchain-based e-voting is generically weaker than confidentiality of classical e-voting protocols, in the simplified blockchain model.*"

Algorithm 5 Conversion of Smart Contracts

Input: $V(read_p(t), append_p(r), contract)$
Output: A public bulletin board based protocol, with additional actions for public

1: $BB \leftarrow \begin{cases} R_{BB} = contract.initialState \\ read'_P : T \rightarrow R \cup \{\perp\} \\ write_P : R \rightarrow T \cup \{\perp\} \\ delete_P : T \rightarrow \{\top, \perp\} \end{cases}$
2: **for** contract function f in *contract* **do**
3: $f' \leftarrow E_{BB}(contract).f.$
4: $BB.write_p$(Complete description of f')
5: Replace each call of f in V into:
6: Procedure {
7: p requests CA to trigger f'.
8: $BB.write_{CA}(f'||p)$
9: $BB.write_{CA}$(State of $E_{BB}(contract)$)
10: $result \leftarrow f'()$
11: **if** $result = \perp$ **then**
12: $BB.write_{CA}$("Failure")
13: **else**
14: $BB.write_{CA}$(State of $E_{BB}(contract)$)
15: **end if**
16: }
17: **end for**
18: **return** $V(read'_p(t), write_p(r))$

The public bulletin board, which is produced after the transformation, may require honesty of the CA as the manager. Corruption of the CA does not degrade the confidentiality, since modification and deletion of records posted on the bulletin board only drops the amount of knowledge of each parties on the records. A malicious CA may post some personal knowledge; the CA is also capable of posting the same knowledge even on blockchain-based voting protocols.

6 Verifiability Requirements

6.1 Universal Verifiability

Universal Verifiability states that an outsider should be able to verify whether the final result is derived from public voting records. Universal verification is done by the public, using public records including ballots cast by voters. As a necessary condition, public data should not be modified by the attackers to ensure universal verifiability. We cannot simply make transformations from blockchain to non-blockchain protocols just like prior arguments, because they introduce additional assumptions on the honesty of the CA who manages the bulletin board. Obviously, a malicious CA breaking this assumption may degrade the integrity significantly. From the perspective of blockchain as a decentralized public bulletin board, the following 'inverse' transformation safely removes the CA managing the bulletin board who might be adversarial.

Table 1: Construction of emulator E_{BB}

Smart Contract Instructions	Emulator's Instructions
Arithmatic and logic operations	No modifications
Control flow	No modifications
Stack, memory, and storage instructions	Converted to invocation of BB's oracles, while maintaining storage in BB
Block information query	Conversion to *read* oracles in BB
Environmental instructions	(Unused in record-independent protocols)
Use of Public Keys	Mapped into corresponding entities

Algorithm 6 Conversion to Simplified Blockchain

Input: $V(read_p(t), write_p(r), delete_p(t))$
Output: A blockchain based protocol

1: $BC \leftarrow \begin{cases} R_{BC} = \phi \\ read'_P : T \rightarrow R \cup \{\perp\} \\ append_P : R \rightarrow T \cup \{\perp\} \end{cases}$
2: $b(t) \leftarrow (t = \perp)?\perp : \top$
3: $del_p(t) \leftarrow b(append_p(invalidate(read_p(t))))$
4: **return** $V(read'_p(t), append_p(r), del_p(t))$

Therefore, one may conclude that the universal verifiability of blockchain-based e-voting protocols is *generically stronger* than classical e-voting protocols.

6.2 Individual Verifiability

Individual verifiability states that a voter should be able to verify whether his/her ballot is correctly included in tallying procedures. Individual verification is done by the voters, using both public records and their private knowledge.

Cortier and Lallemand [6] show that privacy implies individual verifiability for piecewise-tallying e-voting protocols. As the authors claim that piecewise-tallying "is satisfied by most voting schemes", one may predict that individual verifiability of blockchain-based e-voting protocols is *generically weaker* than classical e-voting protocols.

7 Practical Transformations

7.1 Blockchain as an Append-only Storage

Liu and Wang's protocol [17] involves a voter *Alice*, a vote organizer *Bob*, and an inspector *Carol*. The protocol utilizes a blind signature scheme $\{C, S', C'\}$ satisfying $S'_B(m) = C'_A(S'_B(C_B(m)))$ for all message m and parties A and B, to ensure validity of ballots. The protocol treats blockchain as a ballot box, as well as a storage for public communications. Ballot casting is done by anonymous blockchain accounts to ensure privacy, while communications in ballot preparation are done real-name for eligibility verification.

Algorithm 2 can be applied to let *Alice* cast her vote, using BC and *Alice* as B and i in the algorithm. The statement $B \leftarrow B \cup \{b\}$ can be implemented by $BC.append_{Anonymous}(b)$.

Algorithm 7 $makeBallot_{Alice}(selection_{Alice})$

Input: $selection_{Alice}$
Output: Ballot to cast

1: Create a vote string V with $selection_{Alice}$.
2: $h_V \leftarrow hash(V)$
3: $t \leftarrow BC.append_{Alice}(h_V, C_{Alice}(h_V))$
4: *Alice* notifies t to *Bob* and *Carol*
5: $t_{Bob} \leftarrow BC.append_{Bob}(S'_{Bob}(BC.read_{Bob}(t)))$
6: $t_{Carol} \leftarrow BC.append_{Carol}(S'_{Carol}(BC.read_{Carol}(t)))$
7: *Bob* and *Carol* notify t_{Bob} and t_{Carol} to *Alice*.
8: $s_{Bob} \leftarrow C'_{Alice}(BC.read_{Alice}(t_{Bob}))$
9: $s_{Carol} \leftarrow C'_{Alice}(BC.read_{Alice}(t_{Carol}))$
10: **return** $V||s_{Bob}||s_{Carol}$

Algorithm 8 $makeBallot'_{Alice}(selection_{Alice})$

Input: $selection_{Alice}$
Output: Ballot to cast

1: Create a vote string V with $selection_{Alice}$.
2: $h_V \leftarrow hash(V)$
3: $t \leftarrow BB.write_{Alice}(h_V, C_{Alice}(h_V))$
4: *Alice* notifies t to *Bob* and *Carol*
5: $t_{Bob} \leftarrow BB.write_{Bob}(S'_{Bob}(BB.read'_{Bob}(t)))$
6: $t_{Carol} \leftarrow BB.write_{Carol}(S'_{Carol}(BB.read'_{Carol}(t)))$
7: *Bob* and *Carol* notify t_{Bob} and t_{Carol} to *Alice*.
8: $s_{Bob} \leftarrow C'_{Alice}(BB.read'_{Alice}(t_{Bob}))$
9: $s_{Carol} \leftarrow C'_{Alice}(BB.read'_{Alice}(t_{Carol}))$
10: **return** $V||s_{Bob}||s_{Carol}$

Upon application of the generic transformation in **Algorithm 4**, BC is replaced to a public bulletin board BB, transforming the protocol into **Algorithm 8**. We can assume that BB is managed by either *Bob* or *Carol*. The statement $B \leftarrow B \cup \{b\}$ in **Algorithm 2** can be implemented by $BB.write_{Anonymous}(b)$.

7.2 Smart Contract-based Protocols

A protocol by Hjalmarsson et al. [12] implements the functionalities of e-voting in smart contract form. This protocol is record-independent as the contracts create storages and any contract functions read or modify the contents inside the storage. Therefore, the protocol would be successfully converted through this generic transformation.

This protocol stores voter lists in a Boolean array `voters`, and candidate information in a structure ar-

ray `candidates`. For instance, ballot casting in this protocol is done as follows:

```
function vote(uint candidate) public{
  require(!voters[msg.sender]);
  if(now >
   candidates[candidate].expirationDate){
   revert();
  }
  candidates[candidate].voteCount += 1;
  voters[msg.sender] = true;
}
```

Note that voters request the contract function `vote` increment partial tally of the candidate to vote, stored in `candidates[candidate].voteCount`.

Through the transformation, each invocation of `vote` by p will be converted using **Algorithm 9** under globally defined emulator of `vote` and the storage tags $t1, t2$, and $t3$ described in **Algorithm 10**.

Algorithm 9 Conversion of `vote(candidate)`

1: p requests CA to trigger $vote$.
2: $BB.write_{CA}(vote||p)$
3: $BB.write_{CA}(\text{“Before call:”}\ ||BB.read_{CA}(t1)$
4: $\quad||BB.read_{CA}(t2)||BB.read_{CA}(t3))$
5: $result \leftarrow E_{BB}(contract).vote(candidate)$
6: **if** $result = \perp$ **then**
7: $\quad BB.write_{CA}(\text{“Failure”})$
8: **else**
9: $\quad BB.write_{CA}(\text{“After call:”}\ ||BB.read_{CA}(t1)$
10: $\quad\quad||BB.read_{CA}(t2)||BB.read_{CA}(t3))$
11: **end if**

Algorithm 10 Emulator of `vote(candidate)`

1: $t1 \leftarrow tag(voters[p])$
2: $t2 \leftarrow tag(candidates[candidate].expirationDate)$
3: $t3 \leftarrow tag(candidates[candidate].voteCount)$
4: $E_{BB}.vote \leftarrow$ **function** $(candidate)\{$
5: $\quad$ **if** $BB.read'_{CA}(t1)$ **then**
6: $\quad\quad$ **return** $\perp$
7: $\quad$ **end if**
8: $\quad expire \leftarrow BB.read'_{CA}(t2)$
9: $\quad$ **if** $now() > expire$ **then**
10: $\quad\quad$ **return** $\perp$
11: $\quad$ **end if**
12: $\quad ct \leftarrow BB.read'_{CA}(t3) + 1$
13: $\quad BB.rewrite_{CA}(t3, ct)$
14: $\quad BB.rewrite_{CA}(t1, true)$
15: $\quad$ **return** $\top$
16: $\}$

8 Discussion

8.1 General Requirements

We have compared confidentiality and verifiability, which are the two key security requirements for e-voting, to analyze benefits and inferior points of blockchain usage in e-voting. However, more requirements of e-voting exist which are rather non-cryptographic. As the generic transformations do not give reasonable comparisons to non-cryptographic requirements, we qualitatively compare the requirements according to the properties of blockchain.

We expect blockchain can enhance availability of e-voting protocols, since a centralized e-voting system has a single point of failure, the central server. Fault-tolerance is one of the strong points of distributed systems such as blockchain. Unless large-scale attacks on blockchain are deployed such as Apostolaki et al.'s partitioning attack [1], one can argue higher availability over centralized configuration without blockchain.

Usability is the willingness of users to use a system or not. Regardless of whether blockchain is used, usability is one of the inherent problems of e-voting, creating discrimination by technology access. Scalability is another common problem of e-voting as experimental evidences are not enough to host large-scale voting, though some protocols claim scalability up to millions of voters.

Different types of voting are used in different circumstances. This leads to a different priority of the requirements for the choice of appropriate protocol in a particular situation. For example, boardroom voting revealing choices of voters can be done with less confidentiality, while strong confidentiality is important for presidential elections.

8.2 Future Works

The construction of the emulator on the bulletin board should be more precisely defined with real-world smart contract instructions such as EVM bytecodes. We also leave quantitative estimation or comparison of non-cryptographic requirements for future work.

Eliminating the possibilities of the Sybil attack by a CA during the registration phase is one of the obstacles e-voting protocols must overcome. One may imagine some configurations which may prevent the Sybil attack, such as a situation in which the CA should convince representatives of all advocate parties of the selections in the registration phase, to correctly register people. However, this attack is out of scope in this paper.

9 Conclusion

Based on the generic transformations on the simplified blockchain model, we demonstrate how the adoption of blockchain affects the requirements of e-voting protocols. One can make a classical e-voting protocol having equivalent confidentiality using a protocol with blockchain, meaning that blockchain-based e-voting protocols have *generically weaker* universal verifiability than classical protocols. One can create an e-voting protocol having equivalent universal verifiability using a classical e-voting protocol, meaning that blockchain-based e-voting protocols have *generically stronger* universal verifiability than classical protocols.

Acknowledgement

This work was partly supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. 2017-0-00555, Towards Provable-secure Multi-party Authenticated Key Exchange Protocol based on Lattices in a Quantum World).

References

[1] Maria Apostolaki, Aviv Zohar, and Laurent Vanbever. Hijacking bitcoin: Routing attacks on cryptocurrencies. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 375–392. IEEE, 2017.

[2] Stefano Bistarelli, Marco Mantilacci, Paolo Santancini, and Francesco Santini. An end-to-end voting-system based on Bitcoin. In *Proceedings of the Symposium on Applied Computing*, pages 1836–1841. ACM, 2017.

[3] David Chaum, Aleks Essex, Richard Carback, Jeremy Clark, Stefan Popoveniuc, Alan Sherman, and Poorvi Vora. Scantegrity: End-to-end voter-verifiable optical-scan voting. *IEEE Security & Privacy*, 6(3), 2008.

[4] Michael R Clarkson, Stephen Chong, and Andrew C Myers. Civitas: Toward a secure voting system. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 354–368. IEEE, 2008.

[5] Véronique Cortier, David Galindo, and Mathieu Turuani. A formal analysis of the Neuchâtel e-voting protocol. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 430–442. IEEE, 2018.

[6] Véronique Cortier and Joseph Lallemand. *Voting: You Can't Have Privacy without Individual Verifiability.* PhD thesis, CNRS, Inria, LORIA, 2018.

[7] Véronique Cortier and Cyrille Wiedling. A formal analysis of the Norwegian e-voting protocol. *Journal of Computer Security*, 25(1):21–57, 2017.

[8] Gaby G Dagher, Praneeth Babu Marella, Matea Milojkovic, and Jordan Mohler. BroncoVote: Secure voting system using Ethereum's blockchain. 2018.

[9] Rifa Hanifatunnisa and Budi Rahardjo. Blockchain based e-voting recording system design. In *Telecommunication Systems Services and Applications (TSSA), 2017 11th International Conference on*, pages 1–6. IEEE, 2017.

[10] Freya Sheer Hardwick, Raja Naeem Akram, and Konstantinos Markantonakis. E-voting with blockchain: An e-voting protocol with decentralisation and voter privacy. *arXiv preprint arXiv:1805.10258*, 2018.

[11] Severin Hauser and Rolf Haenni. A generic interface for the public bulletin board used in UniVote. In *E-Democracy and Open Government (CeDEM), Conference for*, pages 49–56. IEEE, 2016.

[12] Friorik P Hjalmarsson, Gunnlaugur K Hreioarsson, Mohammad Hamdaqa, and Gisli Hjalmtysson. Blockchain-based e-voting system. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pages 983–986. IEEE, 2018.

[13] Shahram Khazaei, Björn Terelius, and Douglas Wikström. Cryptanalysis of a universally verifiable efficient re-encryption mixnet. *IACR Cryptology ePrint Archive*, 2012:100, 2012.

[14] Byoungcheon Lee, Colin Boyd, Ed Dawson, Kwangjo Kim, Jeongmo Yang, and Seungjae Yoo. Providing receipt-freeness in mixnet-based voting protocols. In *International Conference on Information Security and Cryptology*, pages 245–258. Springer, 2003.

[15] Kibin Lee, Joshua I James, Tekachew G Ejeta, and Hyoung J Kim. Electronic voting service using block-chain. *Journal of Digital Forensics, Security and Law*, 11(2):8, 2016.

[16] Peter Hyun-Jeen Lee and Siamak F Shahandashti. Theoretical attacks on e2e voting systems. *Real-World Electronic Voting: Design, Analysis and Deployment*, page 219, 2016.

[17] Yi Liu and Qi Wang. An e-voting protocol based on blockchain. *IACR Cryptol. ePrint Arch., Santa Barbara, CA, USA, Tech. Rep*, 1043:2017, 2017.

[18] Patrick McCorry, Siamak F Shahandashti, and Feng Hao. A smart contract for boardroom voting with maximum voter privacy. In *International Conference on Financial Cryptography and Data Security*, pages 357–375. Springer, 2017.

[19] Mike Orcutt. Why security experts hate that blockchain voting will be used in the midterm elections, 2018. Accessed on Oct 11, 2018.

[20] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 223–238. Springer, 1999.

[21] Bin Yu, Joseph K Liu, Amin Sakzad, Surya Nepal, Ron Steinfeld, Paul Rimba, and Man Ho Au. Platform-independent secure blockchain-based voting system. In *International Conference on Information Security*, pages 369–386. Springer, 2018.

[22] Zhichao Zhao and T-H Hubert Chan. How to vote privately using Bitcoin. In *International Conference on Information and Communications Security*, pages 82–96. Springer, 2015.