DATE: A Decentralized, Anonymous, and Transparent E-voting System

Wei-Jr Lai
National Taiwan University
r05922108@cmlab.csie.ntu.edu.tw

Yung-chen Hsieh
National Taiwan University
will@cmlab.csie.ntu.edu.tw

Chih-Wen Hsueh
National Taiwan University
cwhsueh@csie.ntu.edu.tw

Ja-Ling Wu
National Taiwan University
wjl@cmlab.csie.ntu.edu.tw

Abstract—A trusted electronic election system requires that all the involved information must go public. However, it focuses not only on transparency but also on privacy issues. In other words, each ballot should be counted anonymously, correctly, and efficiently. In this work, an effective e-voting system is proposed for voters to minimize their trust in the authority or government. We ensure the transparency of election by putting all messages on the Ethereum blockchain; in the meantime, the privacy of individual voter is protected via an effective ring signature mechanism. Besides, the attractive self-tallying feature is also built in our system, which guarantees that everyone who can access the blockchain network is able to tally the result on his own, i.e., no third party is required after the voting phase. More importantly, we ensure the correctness of voting results and keep the Ethereum gas cost of individual participant as low as possible, at the same time. Moreover, the pre-described characteristics of stealth address in our system makes it more suitable for large-scale election on line.

Keywords—Electronic Voting System, Ethereum Blockchain, Privacy Preserving, Self-Tallying

I. INTRODUCTION

Although some countries have begun to use electronic voting for elections on a national scale [1], there is still no suitable, trusted, efficient enough electronic voting system for large scale election because it involves many contradictory features. The election needs one or more authorities for both authentication and privacy protection of participants; however, it is difficult for voters to believe in the government or authority that will always obey the voting laws or never get hacked.

There have been many studies and discussions on fair elections [2], for example, Helios opened all ballots to ensure transparency and then permuted the results for anonymity [3]. This approach is still centralized and the tally phase is very time-consuming. Hao et al. proposed an open vote network protocol, which is decentralized, anonymous and transparent [4]. However, it cannot tally the results even only one voter did not cast his ballot, which makes the scheme only suitable for small-scale voting.

With the advances of blockchain technology, some works suggested to use blockchain in electronic voting due to its immutability and public bulletin board characteristics. McCorry ran an open vote protocol [5] on the Ethereum [6] smart contract, which makes the whole process more convenient. More also built their voting systems based on blockchain. Zhao asked all voters to open their masks after voting [7]. Such a requirement causes the same problem as in [4, 5], and it is obviously unreasonable to imperatively force each voter to pay a deposit before voting. Bistarelli et al. proposed a simple and efficient scheme and introduced multiple authorities to protect the privacy of voters [8]. However, it is centralized and not a completely transparent voting system.

In this work, an efficient and effective decentralized, anonymous and transparent voting system is proposed and realized. We ensure the transparency of voting by putting all messages on the Ethereum blockchain and the privacy of each voter by an efficient and effective ring signature mechanism. More importantly, the self-tallying feature is maintained in our system, which makes the voting results can be tallied without the need of a trusted third party. Finally, the effectiveness of our system can be justified by checking the required Ethereum gases per voter.

II. BACKGROUND

A. Ethereum Blockchain

The Ethereum blockchain is an ordered and transaction-based state machine proposed in 2013 and the currency runs on Ethereum is called the ether. The construction of a blockchain relies on miners, who mined a new block via the proof of work (PoW) consensus algorithm. PoW is a computationally complex puzzle which makes the information on blockchain immutable unless the majority of miners are malicious. Different from Bitcoin, there are two types of accounts in the Ethereum blockchain:

- Externally owned account: Public-private key pairs are generated for allowing users to send transactions to off-chain counterparts with specific addresses.
- Contract account: Sets of functions are deployed by users through a transaction, in which the functions are controlled by written codes rather than users directly.

On Ethereum, the main fields of a transaction include:

978-1-5386-4870-4/18/$31.00 ©2018 IEEE

- Source-from: A signature comes from an exterior-owned account, allowing others to verify the correctness of a transaction.
- Destination-to: It contains an address that can be either from an exterior-owned account or a contract account.
- Data-field: It contains the compiled contract codes or instructions for the contract, or we can also store data on it as records.
- Gas price: The exchange rate between ether and gas.
- Gas: The cost that senders should pay to miners, which is usually proportional to the complexity or amount of computation associated with the involved instructions, caused by the transaction.

The state of current Ethereum blockchain is kept in an Ethereum virtual machine (EVM), which runs on each miner's computer. The states of EVMs will be the same as long as the block contents of all miners are identical. Thus, to record data on smart contract is equivalent to change the state of every miner's computer, and this process will cost much higher than an ordinary transaction. Ordinary transaction is a pay-only transaction which won't call any specific function but only pay ethers to other accounts. There are only non-executable data (i.e., no executable codes and/or instructions) stored on the Data-field of an ordinary transaction. Therefore, it will not change the state of blockchain, but its data is still visible as part of the blockchain history.

For accomplishing a transparent and decentralized e-voting, we deployed a smart contract with specific functions on Ethereum. The called instructions are executed according to the embedded codes on the smart contract, and the transparency and safety of the whole voting process are provided by the Ethereum network. By moving data and computation to smart contract, everyone interested in the election can openly access the related information via Ethereum blockchain. In other words, the whole election is decentralized and trusted.

B. Ring signature

The most desirable feature of an electronic election system is Anonymous. The relationships between voters and their ballots must not be revealed in public. Thus, we need a special digital signature scheme called ring signature [9], which was first proposed by Ron Rivest, Adi Shamir, and Yael Tauman in 2001. Ring signature has the property that a signer in a particular group is able to sign the message, as a member of the group, but verifier cannot distinguish the identity of the signer from the other members of the group. We can simply apply this method to make a voter sign his ballot anonymously, but there is a challenging problem: a voter may cast his ballot more than once. This is similar to the double-spending problem in the Bitcoin blockchain.

Thus, we included the one-time ring signatures (OTRS) scheme, proposed by Nicolas van Saberhagen [10], into our system. OTRS scheme ensures that a voter with one key-pair can only sign a ballot once, while the ballot will be marked as (or verified to be) signed by a particular group. The notations and the operating procedures of OTRS scheme are addressed as follows:

The parameters of an OTRS scheme include:

- $\boldsymbol{q}$: a prime number
- $\boldsymbol{E}$: an elliptic curve
- $\boldsymbol{G}$: a base point on $\boldsymbol{E}$
- $\boldsymbol{l}$: a prime order of the base point $\boldsymbol{G}$
- $\boldsymbol{H_s}$: a cryptographic hash function which maps a binary sequence with arbitrary length to a finite field $\boldsymbol{F}$ module $\boldsymbol{q}$; $\boldsymbol{H_s}: \{\boldsymbol{0},\boldsymbol{1}\}^* \rightarrow \boldsymbol{F_q}$
- $\boldsymbol{H_p}$: a cryptographic hash function which maps finite field points on an elliptic curve to themselves; $\boldsymbol{H_p}: \boldsymbol{E}(\boldsymbol{F_q}) \rightarrow \boldsymbol{E}(\boldsymbol{F_q})$
- $\boldsymbol{H_b}$: a cryptographic hash function which maps a binary sequence with arbitrary length to a point on an elliptic curve; $\boldsymbol{H_b}: \{\boldsymbol{0},\boldsymbol{1}\}^* \rightarrow \boldsymbol{E}(\boldsymbol{F_q})$

(1) Generation Step

Assume there is a set of public keys $\{ P_i , i \in [1, n] \}$, and a legal signer owns a private key x_s corresponding to the public key P_s,where $s \in [1, n]$. First, the signer computes another public key $I = x_s H_p(P_s)$ called the "key image", then applies the following transformations:

$$L_i = \begin{cases} q_i G & , \quad if \ i = s \\ q_i G + w_i P_i & , \quad if \ i \neq s \end{cases}$$

and

$$R_i = \begin{cases} q_i H_p(P_i) & , if \ i = s \\ q_i H_p(P_i) + w_i I & , if \ i \neq s , \end{cases}$$

where q_i and w_i are random numbers selected from $[1, \ldots, l]$, the signer then computes the following non-interactive challenge

$$c = H_s(m, L_1, \ldots, L_n, R_1, \ldots, R_n).$$

Finally, the signer computes

$$c_i = \begin{cases} w_i & , if \ i \neq s \\ c - \sum_{i=1}^{n} c_i \ mod \ l & , if \ i = s \end{cases}$$

$$r_i = \begin{cases} q_i & , \ if \ i \neq s \\ q_s - c_s x & , \ if \ i = s \end{cases}.$$

Then, the generated one-time ring signature becomes

$$\sigma = (I, c_1, \ldots, c_n, r_1, \ldots r_n) .$$

(2) Verification Step

Anyone of the verifiers can compute the transformations

$$\begin{cases} L_i' = r_i G + c_i P_i \\ R_i' = r_i H_p(P_i) + c_i I \end{cases}, \ i \in [1, n]$$

then check if the following equation is true or not:

$$\sum_{i=1}^{n} c_i = H_s(m, L_1', \ldots, L_n', R_1', \ldots R_n').$$

C. Stealth Address

With the aid of OTRS scheme, a voter can cast his ballot only once, among a particular set of voters. But there is still one more requirement in an electronic election system, that is the information about the ballot cannot go public until the tally phase. Thus, the voting related messages must be encrypted before being sent out.

To achieve this goal, we also applied the unlinkable payment scheme, proposed in [10], to our system, which allows senders to generate different destinations based on the same public key. In other words, once we can uniquely map the representation of candidate to a coordinate on an elliptic curve with $\boldsymbol{H_b}$, voters can send transactions to different addresses even though they cast to the same candidate. This is called the "stealth address" property of a voting system.

Stealth Address implies that voters can obfuscate observers about which candidates they actually voted for. Stealth Address can be realized in our system as follows:

- A candidate C_i (or the so-called receiver) has two standard key pairs: (a_i, A_i) and (b_i, B_i) on a selected elliptic curve, where $A_i = a_i \boldsymbol{G}$, and $B_i = b_i \boldsymbol{G}$.
- For a voter (or the so-called sender) who wants to generate a stealth address for a selected candidate C_j: He chooses a random number $r \in [1, l-1]$ and computes $R = r\boldsymbol{G}$, then the corresponding stealth address becomes $SA = \boldsymbol{H_s}(rA_j)\boldsymbol{G} + B_j$, and the ballot associated with the candidate C_j is (SA, R).
- Once the verifier gets the private keys: (a_j, b_j) of C_j and the ballot information (SA, R), he can compute

$$x = \boldsymbol{H_s}(a_j R) + b_j$$
$$x\boldsymbol{G} = (\boldsymbol{H_s}(a_j R) + b_j)\boldsymbol{G}.$$

If the ballot is directed to the corresponding C_j, then

$$x\boldsymbol{G} = (\boldsymbol{H_s}(a_j R)\boldsymbol{G} + b_j)\boldsymbol{G} = \boldsymbol{H_s}(a_j r\boldsymbol{G})\boldsymbol{G} + b_j\boldsymbol{G}$$
$$= \boldsymbol{H_s}(rA_j)\boldsymbol{G} + B_j = SA.$$

However, in this way, there are too many redundant calculations when verifying the ballots. We can further simplify the computation of tally by assigning all candidates' first private key pairs (a_i, A_i) to the same (a, A). then the ballot (SA, R) is voted for the candidate C_j if

$$x\boldsymbol{G} = \boldsymbol{H_s}(rA)\boldsymbol{G} + B_j = SA.$$

Of course, there are other elliptic curve cryptography encryption algorithms like the integrated encryption scheme [11] which might be used, but the concerns about the security of messages (e.g. the selected candidates) are relatively scanty.

D. Key management scheme

In a general election, information about election results will not be opened until the tally process begin. For efficiency consideration, we need more than one people (the so-called key managers) to share the first private key for all candidates. We take Diffie-Hellman key exchange to solve the problem as follows:

(1) Two key managers Alice and Bob pick two random numbers r_a, r_b and announce $r_a\boldsymbol{G}$ and $r_b\boldsymbol{G}$, publicly and separately.

(2) Both of them can compute $r_a\, r_b\boldsymbol{G}$ and make the result public.

(3) After the voting phase, Alice and Bob send r_a and r_b to the contract, separately.

The tuple $(r_a\, r_b,\ r_a\, r_b\boldsymbol{G})$ is the candidates' first shared key-pair. The above key exchange method can be easily extended to multiparty situations and ensure that even only one of the participants is honest (i.e., he or she doesn't collude with the others), the secret will not be revealed in advance.

Of course, we can further import deposit schemes to make participants willing to follow the protocol more; however, this is not a necessary feature of our system. The method we chose does not need to generate the secret before sharing it with others, so our system works without the involvement of a centralized authority. Furthermore, we can ask each key manager to sign his or her key-related message, before publishing it in step (1), to ensure that each one of the key managers has the corresponding private key with respect to the public key that he or she has submitted.

III. Proposed System

To make the whole process transparent and convenient, we deployed our system on an Ethereum smart contract and stored all the necessary information on it.

The smart contract can only access the data within its current state, rather than all information about the history of the blockchain. That is, as long as the data do not need to be calculated in the smart contract, we can save them as pointers pointed to some transaction indexes. We stored only the pointers on a ballot because the gas limit of Ethereum

blockchain cannot afford too complex operations, such as self-tallying at once. Because that process needs a lot of gas-consuming computations for verifying the ring signature scheme.

An election, in our system, as shown in Fig. 1., can be split into three phases and operated with the following five steps:

(1) Hosting authorized voters and candidates.

(2) Key managers build candidates' first shared public key.

(3) Valid voters cast their ballots.

(4) Key managers open their secrets.

(5) Anyone can tally the voting results by himself.

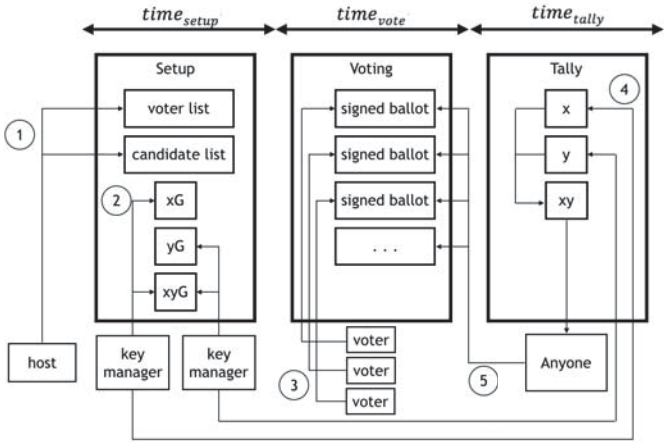

Fig. 1. Block diagram of the proposed DATE Voting System.

A. *Setup phase*

The election related information is announced on an Ethereum smart contract publicly at this phase and it should include the following:

- Voter list: A set of public keys, each one of them is a point on the chosen elliptic curve and corresponds to a valid voter. The public keys of valid voters could be appended on the voter list after they are authenticated. The authentication process can be either centralized or decentralized. The former one can be conducted by a trusted third party. The system assures Anonymity of voters even when the relationships between identity and public key are publicly linkable. This property enables participants to audit the authentication process by checking whether the identity of a voter is in the list or not. The latter one can be realized by integrating the authentication process with the decentralized verification scheme [12], allowing only the voter with specific credential to publish his public key on the list.

- Candidate list: Each candidate can be represented by a string of binary symbols or other data types. In order to generate the shared public key to support the pre-described property of stealth address, each voter can represent each candidate as the coordinates of a point on the chosen elliptic curve [13,14]. As long as voters have reached a consensus on the deterministic hash function, ballots will be counted correctly at the tally phase. Note that we hashed the representation of each candidate C_j to get the aforementioned B_j ; note also that the corresponding b_j is unnecessary in the tally phase.

- Key management scheme: The encryption key can be built by a pre-negotiated set of key managers, as mentioned in Section II.D. All the processes can be publicly conducted on an Ethereum smart contract. For ensuring key managers will submit their individual secret keys (the chosen random numbers) faithfully, they are asked to pay an appropriate amount of ethers as deposits.

- $time_{setup}, time_{vote}, time_{tally}$: The time slots for setup, voting, and tallying phases. Within the period of $time_{setup}$, the smart contract allows participants to complete collecting the above-mentioned information. Then, voters can cast their ballots signed with ring signature during the period of $time_{vote}$. The smart contract does not accept any ballot later than the end of $time_{vote}$. Of course, the smart contract will not start any tallying related process before the beginning of $time_{tally}$, and then key manages can open their secrets to get their deposits back.

B. *Voting phase*

Each valid voter has one of the private keys associated with the public keys stored on the voter list and is aware of the public information including the first public key A of candidates, and of course, the candidate list stored on the Ethereum smart contract. A voter should cast his ballot as follows:

(1) Build a ballot (SA, R) for his selected candidate by computing (A, B_i), where B_i is the hash of the selected candidate C_i, a point on an elliptic curve, actually.

(2) The new ballot is signed with the voter owned private key and the one-time ring signature to get the signed ballot (m, σ), where $m = (SA, R)$.

(3) The voter sends the signed ballot to the smart contract with a different Ethereum account.

Note that if a ballot is sent from the address corresponding to the voter's original private key directly, the anonymity of our voting system will be broken immediately.

Complex operations cost larger amount of gases on an Ethereum smart contract than their plain counterparts. If there is no effect to the correctness of the results, we simplify the required computation as much as possible, based on the techniques presented in [15]. For the same purpose, we separate the ring signature verification process from the voting phase.

As a result, everyone can send invalid or re-voting ballots many times, but those incorrect ballots can always be detected during the tally phase. A rational attacker will not spend gases to launch a fruitless attack.

C. Tally phase

With the given key management scheme, key managers would open their individual secrets for retrieving their deposits and the smart contract won't accept any ballot after the voting phase. Then, those who interested in the result of an election can access all related information and tally the voting results by themselves. They do not need to trust any authority or anyone in the network, and the privacy of every participant is protected by the mechanism of ring signature in the tally phase.

Each valid ballot should be signed with an OTRS. The signed ballot (m, σ) will be counted to candidate C_i if

(1) The key image I never showed before,

(2) $\sum_{i=0}^{n} c_i = \boldsymbol{H_s}(m, L'_0, \ldots, L'_n, R'_0, \ldots R'_n)$,

(3) $\boldsymbol{H_s}(aR)\boldsymbol{G} + B_i = SA$, where the a is the product of key managers' secrets (i.e., $r_a r_b$ in Section II.D).

Restricted by the upper limit of spendable gases per block on Ethereum blockchain, we may not be able to count the ballots on an Ethereum smart contract, directly. However, the same election result will be maintained if everyone on the network reaches a consensus on the ballot counting method, which can be defined on the blockchain before the election.

IV. Experiment and Discussion

A. Time analysis

Our experiments are performed on a MacBook Pro running OS X 10.12.5 equipped with 2 cores, 2.7 GHz Intel Core i5 and 8 GB DDR3 RAM. For communicating with the blockchain conveniently, our code is written in JavaScript.

As shown in Fig. 2, the time spent by voters is mainly in signing the ballot with OTRS, which is linearly proportional to the ring size. The verification time is almost the same as that of the generation counterpart, since the main burden of both is in the computation of the ring signature. For keeping anonymity of voters, we believe that the time spent above is acceptable.

Another advantage of our system is that voters need only few steps. In other decentralized works [5, 7] alike, voters have to pay deposit, submit their commitment to prevent cheating, cast their real ballots, check if everyone should get the deposit back, then open the election result. There are a lot of redundancy, which requires many interactive processes. In contrast, all our participants only have to vote once anonymously. This is a significant saving in time.

B. Gas cost analysis

It is obvious that the data size of a ballot is linearly proportional to the cardinality of the public key set, and a minimum gas cost for each voter is a must for recording information onto the Ethereum blockchain.

To keep data on a smart contract needs much higher gas cost than that on a blockchain. In order to reduce expenses of voters as much as possible, we do not record the ballot information on the Ethereum smart contract directly but save it on ordinary transactions. The ballot information stored on a smart contract is only the index of the ordinary transaction, which allows us to reduce the cost per voter significantly.

As shown in Fig. 3, the gas cost of ordinary transaction is linearly growing up with the size of the public key set, and that of the contract transaction, for sending the index of the ordinary transaction onto contract, is a constant. Our cost of each voter, i.e. the sum of ordinary transaction and contract transaction, is significantly less than that of [5], which requires over two million gas per voter.

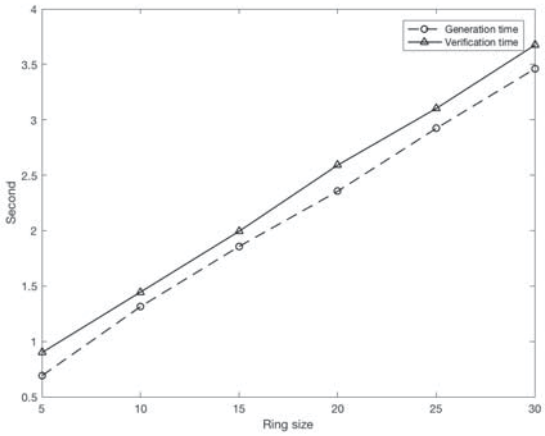

Fig. 2. The required time cost for a ballot generation and verification versus the size of the adopted ring signature.

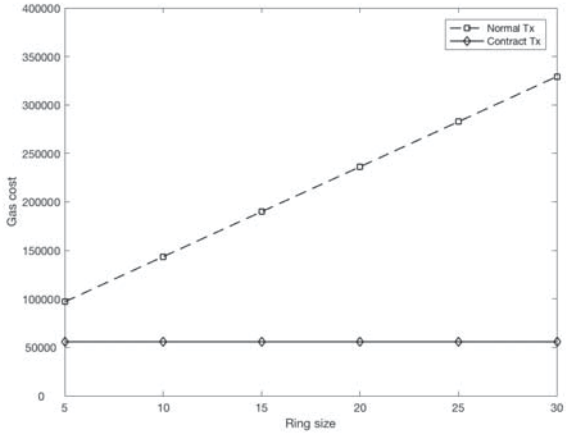

Fig. 3. The gas cost per ballot on an Ethereum smart contract versus the size of the adopted ring signature.

We can further reduce the required cost by saving information on ordinary transactions on IPFS [16], which is a peer-to-peer distributed file system. Voter can save his ballot on IPFS as a file, saving on ordinary transaction expenditure. That is, the ballots we saved on the smart contract are pointers to particular files on IPFS. In other words, voters need only spending gas for contract transaction, which is a constant, in our system. That is, voters can choose the anonymity parameter they want without the need to consider the associated gas cost.

C. Security analysis

The security issues of our system can be split into two parts: the anonymity of voters and the encryption of the selected candidate. The former is protected by using the ring signature mechanism proposed by CryptoNote, which had been adopted by some existing cryptocurrencies, like Monero [17]. There are some studies for analyzing the traceability on Monero [18, 19], which showed the ring signature mechanism can provide certain degree of privacy protection, if the ring size is reasonably large. The temporal analyses conducted in [19] do not work for our system because the transaction output is not related to the input, and the zero mix-ins problem mentioned in [18] can be eliminated by tallying the ballots with a large enough ring size. This implies the anonymity of voters, in our system, is achieved at an acceptable but relatively high complexity. Although this is a necessary evil for protecting the privacy of voters; however, a better way to simplify the privacy protection process is, of course, one of our future research directions.

As for the second part, the encryption scheme we applied is on the basis of the widely-adopted Diffie-Hellman exchange mechanism, which builds a shared secret from the selected random numbers and the public key of the recipient on the chosen elliptic curve.

In the key management scheme mentioned in Section II.D, another risk raises from the collusion of key managers. Although we can increase the difficulty of colluding by increasing the member of key managers, it is still possible to launch such a kind of attack. In the worst case, all of key managers may collude with one another to retrieve the shared secret key. However, the above-mentioned attack can only make the tally phase be executed earlier. Neither can the attackers invade the anonymity of voters nor change the voting results.

V. CONCLUSION

A decentralized anonymous transparent e-voting system is proposed and realized in this work, which requires only minimal trust among participants and as little as Ethereum gas cost per voter, and minimize the interaction steps that a voter requires. By putting all the information on the Ethereum blockchain, the whole election process is transparent and all participants possess identical information. In the tallying phase, since each ballot is independent of one another, our system is free from single point of failure. We do not need to deal with voters who do not want to vote or force them to vote through an additional deposit scheme. Last but not least, since the computations for vote tallying can be sped up via parallelization, thousands of ballots can be verified effectively. Therefore, we believe that the proposed DATE voting system is applicable to a large-scale online election.

VI. ACKNOWLEDGMENT

This work is supported by the Ministry of Science and Technology, Taiwan, Republic of China, under the contract number MOST 106-3144-E-002-009 and 106-2221-E-002 -226.

REFERENCES

[1] Priit Vinkel, "Internet Voting in Estonia." Nordic Conference on Secure IT Systems. Springer, Berlin, Heidelberg, 2011.

[2] Patricia Pesado, et al. "Experiences with Electronic Vote: Challenges and Solutions." *Proceedings of the 9th International Conference on Theory and Practice of Electronic Governance*. ACM, 2016.

[3] B. Adida. Helios: Web-based open-audit voting. In USENIX Security Symposium, volume 17, pages 335–348, 2008.

[4] F. Hao, P. Y. Ryan, and P. Zielinski. Anonymous voting by two-round public discussion. IET Information Security, 4(2):62–67, 2010.

[5] Patrick McCorry, Siamak F. Shahandashti, and Feng Hao. "A smart contract for boardroom voting with maximum voter privacy." *International Conference on Financial Cryptography and Data Security*. Springer, Cham, 2017.

[6] Gavin Wood. "Ethereum: A secure decentralised generalised transaction ledger." Ethereum Project Yellow Paper 151 (2014): 1-32.

[7] Zhichao Zhao, and T-H. Hubert Chan. "How to vote privately using bitcoin." *International Conference on Information and Communications Security*. Springer, Cham, 2015.

[8] Stefano Bistarelli, et al. "An end-to-end voting-system based on bitcoin." *Proceedings of the Symposium on Applied Computing*. ACM, 2017.

[9] Ronald L. Rivest, Adi Shamir, and Yael Tauman. "How to leak a secret." International Conference on the Theory and Application of Cryptology and Information Security. Springer, Berlin, Heidelberg, 2001.

[10] Nicolas van Saberhagen, "Crypto Note v 2.0." CryptoNote. org.[Online] 17.10 (2013).

[11] V. Gayoso Martínez, L. Hernández Encinas, and C. Sánchez Ávila. "A survey of the elliptic curve integrated encryption scheme." *ratio* 80.1024 (2010): 160-223.

[12] Christina Garman, Matthew Green, and Ian Miers. "Decentralized Anonymous Credentials." *NDSS*. 2014.

[13] Dan Boneh, Ben Lynn, and Hovav Shacham. "Short signatures from the Weil pairing." *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, Berlin, Heidelberg, 2001.

[14] Thomas Icart, "How to hash into elliptic curves." *Advances in Cryptology-CRYPTO 2009*. Springer, Berlin, Heidelberg, 2009. 303-316.

[15] Ting Chen, et al. "Under-optimized smart contracts devour your money." Software Analysis, Evolution and Reengineering (SANER), 2017 IEEE 24th International Conference on. IEEE, 2017.

[16] Juan Benet, "IPFS-content addressed, versioned, P2P file system." *arXiv preprint arXiv:1407.3561* (2014).

[17] Monero - secure, private, untraceable, https://getmonero.org

[18] Adam Mackenzie, Surae Noether, and Monero Core Team. "Improving Obfuscation in the CryptoNote Protocol." (2015).

[19] Amrit Kumar, et al. "A traceability analysis of monero's blockchain." *European Symposium on Research in Computer Security*. Springer, Cham, 2017.