Journal of Information Security and Applications 59 (2021) 102815

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

A secure end-to-end verifiable e-voting system using blockchain and cloud server

Somnath Panja *, Bimal Roy

Applied Statistics Unit, Indian Statistical Institute, Kolkata 700108, India

ARTICLE INFO

Keywords:
E-voting
Blockchain
Ethereum
Direct-Recording Electronic
Authentication
Non-interactive zero-knowledge proof
End-to-end verifiable
Secure multi-party computation

ABSTRACT

We propose a cryptographic technique for an authenticated, end-to-end verifiable and secret ballot election. Currently, almost all verifiable e-voting systems require trusted authorities to perform the tallying process except for the DRE-i and DRE-ip systems. We have shown a weakness of the DRE-ip system and proposed a solution. We propose a secure and verifiable voter registration and authentication mechanism. The proposed scheme prevents ballot stuffing attack. We have modified the DRE-ip system so that no adversary can create and post a valid ballot on the public bulletin board without detection. We propose a method for publishing the final tally without revealing the tally from individual Direct-Recording Electronic (DRE) machines using secure multi-party computation and non-interactive zero-knowledge (NIZK) proof. We propose two methods to store these ballots using blockchain and cloud server. To the best of our knowledge, it is the first end-to-end verifiable DRE based e-voting system using blockchain. We provide security proofs to prove the security properties of the proposed scheme. We prove that the efficient NIZK proof proposed by Lin et al. in APSIPA ASC 2019 is not correct since it does not satisfy the witness indistinguishability property of a zero-knowledge proof. We introduce an improved NIZK proof that boosts the efficiency of the system. The experimental data obtained from our tests show the protocol's potential for real-world deployment.

1. Introduction

Election is a process of establishing the democracy in the country. It is also one of the most challenging task, one whose constraints are remarkably strict. Each voter should receive assurance that her vote is *cast as intended*, *recorded as cast* and *tallied as recorded*. In addition, the election system as a whole should ensure that voter coercion is unlikely, even when voters are willing to be influenced. There has been extensive adoption of Direct-Recording Electronic (DRE) devices for voting at polling stations around the world. Starting with the seminal work by Chaum, published in IEEE Security & Privacy [1] in 2004, research on end-to-end (E2E) verifiable e-voting has become a thriving field. Informally, the notion of E2E verifiability refers to have three properties: (1) each voter is able to verify if her vote has been cast as intended; (2) each voter is able to verify if her vote has been recorded as cast; (3) anyone can verify if all the votes have been tallied as recorded. By contrast, in traditional paper-based voting system, a voter cannot verify how her vote is recorded and tallied in the voting process. As with traditional elections, voters go to their polling station, prove their eligibility for casting votes by presenting their identity card. The voter is given a token [2] that allows her to cast vote for her chosen candidate. Therefore, the system depends on trustworthy individual at the polling stations, thus leading to the introduction of automated paperless secure e-voting system. In this paper, we propose a secure authenticated DRE based E2E verifiable e-voting system without tallying authorities.

Hao et al. proposed a voting system, called DRE-i (DRE with integrity) [3], to achieve E2E verifiability without involving any tallying authorities (TAs). However, the pre-computation strategy requires that the pre-computed data is securely stored and accessed during the voting phase. This introduces the possibility for an adversary to break into the secure storage module and compromise the privacy of all ballots. To overcome this issue, Shahandashti et al. provided a voting system, called DRE-ip [4] (DRE-i with enhanced privacy). DRE-ip achieves E2E verifiability without TAs and simultaneously a significantly stronger privacy guarantee than DRE-i. However, both DRE-i and DRE-ip systems necessitate the requirement of a secure append-only public bulletin board (BB). If the BB or the voting machine or the private key of the signature is compromised, an attacker can change some ballots and add additional ballots as well in such a way that it cannot be detected by the DRE-ip tally verification algorithm. The private key of the signature might be compromised at the setup stage.

* Corresponding author.
E-mail addresses: somn.math2007@gmail.com (S. Panja), bimal@isical.ac.in (B. Roy).

https://doi.org/10.1016/j.jisa.2021.102815

Available online 23 April 2021
2214-2126/© 2021 Elsevier Ltd. All rights reserved.

In his PhD thesis, Benaloh [5] assumes BBs with a secure append-only write operations, also stressing out that "implementing such bulletin boards may be problem unto itself". Although the assumption that the BB is a trusted centralized entity is common in the literature, the importance of removing the BB as a single point of failure has been extensively discussed in the recent works of Culnane and Schneider [6], Chondros et al. [7] and Kiayias et al. [8]. In [8], Kiayias et al. show a weakness of the bulletin board proposed in [6] and improve the system. However, in [8], for n peers, the minimum number of honest item collection peers that receive and store submitted items must be greater than $2n/3$ to ensure correct behavior of their bulletin board design. In [9], Küsters et al. raise awareness of an attack, which they call a clash attack, on the verifiability of some of the well-known e-voting systems (for example, ThreeBallot and VAV voting systems [10], a variant of the Helios voting system [11] and the Wombat Voting system [12]). Küsters et al. show that, if the voting machine and the bulletin board collaborate, a bulletin board can replace some ballots by its choice so that it cannot be detected. In [13], Benaloh et al. describe their Trash attack on some well-known verifiable e-voting systems if the bulletin board is compromised.

Instead of assuming a secure append-only public bulletin board, we have modified the DRE-ip algorithm to make it tamper-evident and have proposed two methods (depending on how the election is arranged) to store the ballots using blockchain and cloud server. Since the inception of Bitcoin [14] in 2008, researchers have proposed numerous blockchain-enabled solutions for problems in areas such as artificial intelligent, big data management [15], internet of things [16], 5G [17], health-care [18] and shipping [19] etc. A comprehensive literature review of blockchain-based solutions can be found at [20]. We have measured the costs in terms of Ethereum Gas (and US dollars) to verify and store each ballot on Ethereum blockchain. In this case, all the ballots and public keys of the system remain tamper-resistant. This system prevents coercion even when voters are willing to be influenced. For example, voters may collude with an adversary to vote in favor of adversary's chosen candidate and may intend to prove their choice of vote after the voting process. Our proposed system uses the exponential ElGamal cryptosystem to encrypt a vote. The system generates two distinct generators of the group whose logarithmic relationship is unknown. The system securely deletes the random variable and the vote ('confirmed' vote) for each voter. Consequently, the voter cannot prove her chosen candidate to the adversary. Thus, this kind of coercion can be avoided. Normally, some kinds of coercion-resistance are also provided by the supervised environment of the polling station voting [21]. In addition, since each DRE machine normally covers voting process for small regions, revealing the tally from each DRE machine discloses the distribution of voters against various political parties in small regions. Disclosure of this distribution may impact financial investments, economic development and social security of those small regions. This is also a breach of voter's privacy to some extent. We propose a secure multi-party computation scheme with non-interactive zero-knowledge (NIZK) proof to compute the final tally correctly while keeping the tally from each DRE machine secret.

We also propose a novel method for voter registration and authentication in a verifiable manner using Fuzzy Vault algorithm [22]. As opposed to the traditional biometric based authentication systems, we do not store the biometric template (fingerprint) of individual voters. There are some privacy and security advantages of our proposed voter registration and authentication system. First, we store a biometrically encrypted key corresponding to an individual voter from which neither the biometric nor the key can be retrieved. The secret key itself is independent of the biometric and can be changed or modified. Secondly, we do not rely on fingerprint alone for authentication since people leave fingerprint everywhere inadvertently. In addition, there may be Mafia-owned businesses that collect fingerprint data in large quantities if there is any exploit path. Thirdly, we present a two factor authentication scheme relying on fingerprint of the voter and a smart card (containing a secret key) given to the voter. Fourthly, our scheme is publicly verifiable. We show that the correctness of our system is verifiable by the public. As a result, the system thwarts ballot stuffing attack.

Our contributions. Our contributions in this article include the following.

1. We have shown one weakness of the DRE-ip system. An attacker can post ballots in such a way that it cannot be detected by the tally verification process. We have proposed a solution to prevent this attack.
2. We propose a secure and verifiable voter registration and authentication mechanism using voter's biometric information (fingerprint). The proposed system prevents the well-known ballot stuffing attack.
3. We also propose a method for publishing the final tally when multiple DRE machines are used in a regional zone keeping the result (i.e. tally) from each DRE machine secret. By a regional zone, we mean a constituency from where a candidate is to be elected such as a district instead of the whole country. Thus, our system hides the distribution of voters against various political parties in small areas where DRE machines are used.
4. Depending on how the election is organized, we propose two methods to store the ballots on a public bulletin board.
5. To the best of our knowledge, it is the first end-to-end verifiable DRE based e-voting system using blockchain.
6. We prove that our scheme satisfies the eligibility verifiability property. We also provide security proofs to show that the proposed protocol is end-to-end verifiable as well as preserves each voter's privacy and the integrity of the system.
7. We prove that the efficient NIZK proof algorithm proposed by Lin et al. [23] is not correct since it does not satisfy the properties of a zero-knowledge proof. We propose an efficient 1-out-of-n NIZK algorithm (i.e. the prover Algorithm 3 and the verifier Algorithm 4) involving conjunction and disjunction of multiple assertions. The security proofs of the proposed efficient NIZK proof algorithm are given in Appendix B.
8. We have analyzed the performance of our scheme involving our proposed NIZK proof systems and compared it with existing protocols.

2. Related work

There has been extensive research on e-voting system over the past two decades. Researchers have proposed a number of E2E verifiable schemes and some of these are used in practice. Notable E2E e-voting systems include Votegrity [1] (proposed by Chaum), Markpledge [24], Prêt à Voter [25], STAR-Vote [26], Punchscan [27], Scratch & vote [28], Scantegrity, Scantegrity II [29], Helios [11], Bingo Voting [30], Wombat [12], DRE-i [3], DRE-ip [4]. A review of these systems can be found in [31]. Many other schemes follow similar approaches, in particular, a variant of Prêt à Voter, vVote, was used in 2014 state election in Victoria, Australia [32]. Scantegrity [29] was trialled in local elections in Takoma Park, Maryland, USA [33]. Helios [11] was used to elect Universitè catholique de Louvain in 2009 and it has been used in universities and associations (IACR and ACM). Other schemes that have been used in internal university or party elections include Punchscan [27], Bingo Voting [30], Wombat [12] and DRE-i [3]. However, almost all DRE based E2E verifiable systems require a secure bulletin board. Our system relaxes the requirement of secure BB and provides efficient solution using blockchain and cloud server. All of the above systems consider voter registration and authentication outside of their scope. In our proposed system, we have incorporated a secure and verifiable voter registration and authentication mechanism using biometric.

There are few online internet voting systems based on blockchain. In [34], Zhao et al. proposed a voting system using Bitcoin [14]. In their voting system, the vote does not need to be encrypted and decrypted. Random numbers are used to hide the ballot that are distributed using zero-knowledge proof. Tarasov et al. proposed a cryptocurrency based voting system in [35]. This system is based on the payments that she receives from the voter. The problem with this system is that malicious voters may refuse to "pay" the candidate to retain the money. Furthermore, a centralized trusted authority who coordinates between the candidates and voters must exist. There are smart contract based internet voting systems [36] based on Open Vote Network (OVN), which only support two candidates ("YES"/"NO" voting) and voting is restricted to limited (approximately 50) participants. Seifelnasr et al. [37] have proposed improvements to OVN, introducing an off-chain untrusted administrator to perform the bulk computations. Tivi [38], Followmyvote [39] and The Blockchain Voting Machine [40] are commercial internet voting systems that use the blockchain as ballot box. They claim to achieve verifiability and accessibility anytime and anywhere; however, the voter's privacy in these systems are hard to evaluate. Recently, a blockchain-based voting service has been launched by the Abu Dhabi Securities Exchange [41]. In February 2016, Nasdaq, in cooperation with the Estonian Government, announced a blockchain-based e-voting system for shareholder voting [42,43]. In Estonia, a blockchain-based e-voting system has also been proposed for internal elections of political parties. In a report [42] by Scientific Foresight Unit of the European Parliamentary Research Service, the possibility of using blockchain in e-voting systems has been discussed. Compared with these e-voting systems, ours does not depend on any tallying authorities, and the tallying integrity can be verified by the public. In addition, we propose a secure voter authentication mechanism using biometric (fingerprint) and prove that our scheme satisfies eligibility verifiability property. In [44], Gaudry et al. showed two attacks on the encryption scheme used in the Moscow internet voting system, and in the first attack the authors showed that the used key sizes were too small.

3. Preliminaries

In this section, we focus on the trust requirements and cryptographic assumptions based on which we prove the security properties of our proposed protocol.

3.1. Trust requirements

We describe our trust requirements that our scheme is expected to meet.

- **Voter registration official.** Each voter's biometric (for example, fingerprint) template is collected during the voter registration phase so that only eligible voters are registered with rights to vote. A biometric encryption of the collected biometric data is securely stored by the election authority. Voter registration information is then displayed on a public bulletin board. The voter should check her registration data against her voter ID number (and name) on the public bulletin board using her receipt (that will be provided during the voter registration phase). Voters should raise an issue if their receipt does not match with those on the public bulletin board.
- **Key management for mixnet server.** We use mixnet servers to display all voters' registration data in a public bulletin board. We assume that at least one mixnet server is honest and does not reveal its secret key. The secret keys of the mixnet servers can be distributed among officials using a threshold cryptography.
- **Integrity.** We assume that the voting machine or the BB may alter the voter's vote or change the tallying results. It may happen by accident (for example, due to some software bugs) or by malice (for example, due to some adversarial attack). However, we require that any such changes will be detected even when the machine is completely controlled by an adversary. We prove that our scheme meets this requirement.
- **Vote secrecy.** When a voter selects her chosen candidate on the touch screen, the DRE machine learns her vote by definition. This is inevitable. We assume that the DRE machine keeps the voter's choice secret. However, we require that when the DRE machine is momentarily compromised by an adversary, the adversary will only learn the partial tally at the time of compromise and the vote currently being cast, but nothing beyond that. We also prove that our scheme satisfies this requirement.

3.2. Assured properties

The following properties should be fulfilled by an e-voting protocol. We prove that the our scheme also satisfies the following properties.

- **Eligibility verifiability.** An observer can verify that those voters who have completed the registration phase successfully can only cast their ballots and there is at most one vote per one voter [45].
- **End-to-end verifiability.** As mentioned before, a voting scheme is end-to-end verifiable if it has a mechanism to ensure that every ballot is cast as intended, recorded as cast and tallied as recorded.
- **Voter's privacy.** Over the years, researchers have refined the notion of voter's privacy into following three properties [31].
  1. Ballot secrecy: The voting system must not reveal the vote which an encrypted ballot corresponds to.
  2. Receipt-freeness: The voting system should not provide the voter any evidence to prove to a third party how she voted.
  3. Coercion-resistance: The voter should be able to cast her vote in favor of her chosen candidate even when she appears to be cooperating with a coercer.

3.3. Cryptographic assumptions

We first describe some notations that we use throughout our paper.

Notation. We use the same notations that are used in the DRE-ip system. These notations were introduced by Camenisch and Stadler [46]. We use $P_K\{\lambda : \Gamma = \gamma^\lambda\}$ to denote a non-interactive *proof of knowledge* of a secret λ such that $\Gamma = \gamma^\lambda$ for publicly known Γ and γ. We shorten the notation to $P_K\{\lambda\}$ where context is clear. We use $P_{WF}\{A : X, \ldots, Y, Z\}$ to denote a *proof of well-formedness* of A with respect to $X, \ldots, Y, Z$. We shorten the notation to $P_{WF}\{A\}$ where context is clear.

Cryptographic setup. Our proposed system works over an ECDSA like group setting or a DSA like multiplicative cyclic group setting where the decisional Diffie–Hellman (DDH) assumption holds. In particular, we can choose two large primes p and q such that q divides $(p-1)$. Then we choose the subgroup $\mathbb{G}_q$ of order q of the group $\mathbb{Z}_p^*$ and assume that g is the generator of $\mathbb{G}_q$. q must be greater than the number of voters. The decisional Diffie–Hellman assumption [47] is given below. We use the DDH assumption to prove the security properties of our proposed protocol.

Assumption 1 (*DDH*)**.** The two probability distribution $\{(g^a, g^b, g^{ab}) : a, b$ are uniformly and independently chosen from $\mathbb{Z}_q^*\}$ and $\{(g^a, g^b, g^c) : a, b, c$ are uniformly and independently chosen from $\mathbb{Z}_q^*\}$ are computationally indistinguishable in the security parameter $n = \log(q)$.

Unless otherwise specified, some notions are defined in Table 1.

Table 1
Notations.

Notation	Description
M	An upper bound on total number of voters
n	The number of candidates contesting the election, $n(n >= 2)$
v_i	The vote corresponding to ith ballot, the value M^{j-1}, where $j \in \{1, 2, \ldots, n\}$
λ	The secret r_i corresponding to the ith ballot
g_i	A generator of the group $\mathbb{G}_q$, $\forall i \in \{1, 2\}$
U_i	For the ith ballot, $g_1^{r_i}$
V_i	For the ith ballot, $g_2^{r_i} \cdot g_2^{M^{j-1}}$, where $j \in \{1, 2, \ldots, n\}$
Γ'	U_i (i.e. $g_1^{r_i}$) corresponding to the ith ballot
γ'	It represents g_1
Γ_l''	$V_i / g_2^{M^{l-1}}$ corresponding to the ith ballot, where $l \in \{1, 2, \ldots, n\}$
γ_l''	It represents $g_2, \forall l \in \{1, 2, \ldots, n\}$

3.4. Background information

Protocols for secure E2E verifiable e-voting systems rely on various cryptographic primitives or building blocks. In this section, we briefly discuss those cryptographic primitives. Over the years, researchers in the field of e-voting literature use various public key cryptosystems to make their system secure and verifiable. A comprehensive review of various public key cryptosystems and their security properties could be found in the book by Stinson [48]. We use ElGamal encryption and its variation in our proposed e-voting systems.

3.4.1. Biometric encryption

Biometric technologies add a new level of authentication to various applications; however, there are always risks and challenges related to privacy and security of the biometric. Some of technical challenges in biometric authentication algorithms include accuracy, reliability and security of the biometric. The main privacy and security risk related to the biometric technologies is creating a digital artifact of the biometric from the stored biometric template data in such a way that it will match with the original biometric data, called masquerade attack. Some other security and privacy risks include spoofing attack, replay attack, overwriting YES/NO result attack, tampering attack, Trojan horse attack, substitution attack, misuse of the biometric image (data theft), unauthorized secondary uses of the biometric etc. These kinds of risks limits the uses of biometric technologies in practical applications. Biometric encryption technologies can enhance both the security and privacy of the biometric data. In 1996, Tomko et al. [49] first introduced the concept of biometric encryption. A comprehensive review of biometric encryption technologies can be found in papers [50–53]. Biometric encryption is a technique that either binds a randomly generated key with the biometric or generates a key from the biometric. The resulting data is called biometrically encrypted key or biometrically encrypted data. This biometrically encrypted key is then stored. This key can be regenerated only when a correct fresh biomeric is presented.

3.4.2. Zero-knowledge proof

Zero-knowledge proof was first introduced by Goldwasser, Micali, and Rackoff [54] to prove the truth of a statement without conveying any other information. Subsequently, Bellare and Goldreich [55] refined the definition of zero-knowledge proofs to distinguish them from proofs of knowledge. We use Schnorr's proofs of knowledge of discrete logarithm [56]. We then apply the technique proposed by Cramer, Damgård Schoenmakers [57] to construct proof of disjunctive, conjunctive and combination of both. Fiat–Shamir heuristic [58] is applied to make the constructed proof non-interactive. The security proofs are in random oracle model [59]. The index i of the transaction is embedded as input to the hash function to bind the proof to the transaction. In this paper, we propose an efficient NIZK proof. The proposed prover (Algorithm 3) and the verifier (Algorithm 4) algorithms are described in detail in a later section.

3.4.3. Blockchain

Since the invention of Bitcoin, research on bitcoin [14] and blockchain has become a thriving field. One other blockchain that has become highly influential in research community is Ethereum [60]. In this section, we briefly describe blockchain. Blockchain uses its peer-to-peer network to accept and store the transactions in a decentralized fashion. The network stores the transactions by hashing the into the ongoing blockchain. While adding a transactions into the blockchain, it uses a hash-based proof-of-work mechanism to form a record that cannot be changed without redoing the proof-of-work. The longest chain in the blockchain network serves as the proof of sequence of events witnessed. As long as majority of the CPU power in the network is controlled by the honest nodes, they will generate the longest chain outpacing the dishonest nodes. In a blockchain's network, messages are broadcast, nodes can leave or rejoin the network at their will, accepting the longest chain as a proof of what happened while they were gone.

Transaction. Each transaction bears the digital signature of its owner. Transactions are broadcast and all the nodes agree to accept the longest chain of blocks as the current record of the blockchain. Transactions are stored in the order in which they were received into the blockchain. The structure of a transaction is defined by the corresponding blockchain.

Blocks and hashing. A block may contain several transactions arranged in a Merkle tree fashion. The transactions are hashed into a Merkle tree and only the root of the tree is included in the block's hash. A cryptographic collision resistant hash function is used to take hash of the block and published widely. Each hash includes the previous hash in its hash to form a chain with each additional hash reinforcing the previous one, and hence any modification in the chain will be detected by checking the hash values. Each node checks the hashes of all blocks in the blockchain and accepts the longest chain whose hashes match correctly. The network avoids double inclusion of the same transaction or block that has already been added earlier.

Proof-of-work. A proof-of-work is a computationally expensive task that will be performed by a miner to include a transaction or a block into the blockchain. As the blocks are added one after the another into the blockchain after performing proof-of-work and the previous hashes are included into the current block, to modify a prior block in the blockchain, an attacker has to redo the proof-of-work computation for that block as well as all the blocks after it in the blockchain so that it can readjust all the hashes in the blockchain from that block. The exact proof-of-work algorithm depends on the corresponding blockchain. In Bitcoin, the proof-of-work system is similar to Adam Back's hashcash [61]. The proof-of-work involves searching for a value that when hashed begins with a predefined number of zero bits. Therefore, to perform this task, the average work to be done is exponential with the number of predefined zero bits; however, it can be verified performing only a single hash computation. In Bitcoin, this proof-of-work is performed by incrementing a nonce in the block until a value is found such that the block's hash begins with the required number of zero bits. Once a block is appended into the blockchain after performing this proof-of-work, it cannot be changed without redoing the proof-of-work. As new blocks are appended one after the another into the

blockchain, modifying a block requires to perform the proof-of-work task again for that block as well as all the blocks appended after it in the blockchain.

Network. The network is similar to Bitcoin's peer-to-peer network [14].

3.4.4. Mixnet

Consider a set of messages sent by various senders who wish to shuffle the messages without reveling the secret permutation. This functionality was first introduced by Chaum [62] in 1981 and called the protocol as mixnets. Since then, several mixnet protocols are proposed in the literature based on different definitions and constructions. These mixnets can be classified into two types: heuristics-based mixnets and robust mixnets. In heuristics-based mixnets, the shuffling is usually done for low latency applications and executed almost synchronously, for instance, web browsing. This kind of mixnet protocols maintain some level of privacy; however, some mixnet servers may drop or corrupt messages, although the impact of this may not be serious. In this case, a different set of mixnet servers can be used to perform proper mixing. In robust mixnets, the requirements are very strict such as inputs messages cannot be modified or dropped. These mixnets can be used in applications like voting. The shuffling may take significant time like hours or even a day since the mixing is done in large batches. The privacy of the shuffled permutation should also be preserved and should be provably secure and protected. In 1990, Sako and Kilian [63] proposed a new kind of mixnet protocol with a new property, called universally verifiable mixnet. The proof of correct shuffling provided by the mixnets can be verifiable by the public or any observer of the protocol. Since the generation of the proof for the correct shuffling takes a significant amount of time, the efficiency of this mixnet protocol decreased. Generally, these kinds of universally verifiable mixnet protocols publish all its inputs, outputs and the proof of correct shuffling on a public BB so that anyone can verify the correctness of shuffling. Neff [64] proposed a universally verifiable mixnet protocol that requires $8N$ exponentiations (not counting bulk modexp optimizations). To preserve the privacy and the integrity of the system, it is assumed that not all mixnet servers cooperate with the adversary. This means that, to preserve the privacy and the integrity, at least one mixnet server must be honest who does not reveal its secret key and the random permutation to the adversary.

3.4.5. Secure multi-party computation

In 1986, Yao [65] proposed garbled circuit representation to perform any secure multi-party computation. In this method, decomposition of the computation is performed bit-by-bit basis using several gates. Although Yao's protocol can compute any multi-party computation securely preserving the secrecy of their input, it is inefficient in practice. In 1987, Goldreich et al. [66] proposed the GMW protocol to perform secure multi-party computation with honest majority. Ben-Or et al. [67] proposed a secure multi-party computation protocol, called BWG protocol. BWG protocol defines how to compute addition and multiplication on secret shares and is often used with Shamir secret sharing schemes. A comprehensive study on secure multi-party computation can be found in [68].

4. A weakness of the DRE-ip system

In this section, we recall the DRE-ip [4] system, present one weakness of the system and provide a countermeasure. We describe the DRE-ip algorithm almost verbatim as given in [4].

4.1. The DRE-ip system

We describe the DRE-ip algorithm for the case when there are only two candidates contesting the election. The DRE-ip algorithm does not require any tallying authority. Let v_i be the vote corresponding to the ith ballot, then we have $v_i \in \{0, 1\}$. During the setup phase, the algorithm chooses two generators g_1, g_2 of the corresponding group $\mathbb{G}_q$ such that their logarithmic relationship is unknown. The DRE keeps track of the partial sum $s = \Sigma r_i$ for the random numbers r_i generated on the fly and the running tally $t = \Sigma v_i$ for the cast (confirmed) vote v_i. The system incorporates Benaloh-style voter-initiated auditing [69] in which the voter gets option to audit their vote generated by the DRE to get confidence that the DRE generates the ballot according to her vote. An audited ballot cannot be used to cast a vote. At the end of the voting phase, the set of total ballots $\mathbb{B}$ will be comprised of the set of all audited ballots $\mathbb{A}$ and the set of all confirmed ballots $\mathbb{C}$ i.e. $\mathbb{B} = \mathbb{A} \cup \mathbb{C}$.

Voting phase: This phase involves the DRE, voter and the bulletin board:

1. The voter enters the booth, starts voting and keys in her vote $v_i \in \{0, 1\}$.

2. The DRE generates random $r_i \in \mathbb{Z}_q$ and computes $U_i = g_1^{r_i}, V_i = g_2^{r_i} g_2^{v_i}, P_{WF}\{V_i : g_1, g_2, U_i\}$.

The DRE provides a signed receipt including the unique ballot index i and contents of the ballot $U_i, V_i, P_{WF}\{V_i\}$ to the voter.

3. The voter notices that the first part of the receipt is provided, then she chooses to either audit or confirm her vote.

In case of audit:

4. The DRE adds i to the set of audited ballots $\mathbb{A}$ and provides a signed receipt to the voter. The receipt is clearly marked as *audited* including r_i and v_i.

5. The voter takes and keeps the receipt. The voter verifies that v_i reflects her choice. If the verification succeeds, the voting continues from step 1; otherwise if the verification fails, the voter should raise a dispute immediately.

In case of confirmation:

4. The DRE adds i to the set of confirmed ballots $\mathbb{C}$. The DRE updates the tally and the sum: $t = \sum_{j \in \mathbb{C}} v_j$ and $s = \sum_{j \in \mathbb{C}} r_j$.

The DRE provides a signed receipt, clearly marked as *confirmed* to the voter. The DRE securely deletes r_i and v_i.

5. The voter leaves the polling booth with her receipts.

6. The DRE posts all receipts provided to the voter on the bulletin board.

7. The voter matches her receipts with those on the bulletin board to verify that her receipts are posted on the bulletin board.

Tallying phase: This phase involves the DRE, the bulletin board and the public:

1. The DRE posts the final tally t and the sum s on the bulletin board.

2. The public:

- verify that all the well-formedness proofs on the bulletin board are correct (well-formedness verification);

- verify that, for all audited ballots on the bulletin board, U_i and V_i included in the first part of the receipt are consistent with r_i, v_i provided in the second part (along with the system parameters g_1, g_2) (audit consistency verification);

- verify that following equations hold (tally verification):

$$\prod_{j \in \mathbb{C}} U_j = g_1^s \text{ and } \prod_{j \in \mathbb{C}} V_j = g_2^s g_2^t.$$

If at any point during voting phase or tallying phase, any of the verification by the voter or the public fails, the election stuff should be notified. These include step 5 (for the audited votes), step 7 of the voting phase and step 2 of the tallying phase. The proof of well-formedness $P_{WF}\{V_i : g_1, g_2, U_i\}$ can be implemented as $P_{WF}\{V_i\} = P_K\{r_i : ((U_i = g_1^{r_i}) \wedge (V_i = g_2^{r_i})) \vee ((U_i = g_1^{r_i}) \wedge (V_i / g_2 = g_2^{r_i}))\}$. $P_{WF}\{V_i\}$ is a non-interactive zero-knowledge proof of well-formedness of the ballot.

Table 2
DRE-ip bulletin board. $V_j \in \{g_2^{r_j}, g_2^{r_j} g_2\}$, s is the sum of all random variables of confirmed ballots and t is the final tally.

Initial: g_1, g_2	
$i: U_i, V_i, P_{WF}\{V_i\}$	audited, r_i, v_i
...	...
$j: U_j, V_j, P_{WF}\{V_j\}$	confirmed
Final: t, s	

Table 3
Additional ballots added by an adversary.

Initial: g_1, g_2	
$j+1: U_1, V_1, P_{WF}\{V_1\}$	confirmed
$j+2: U_2, V_2, P_{WF}\{V_2\}$	confirmed
...	...
$j+k: U_k, V_k, P_{WF}\{V_k\}$	confirmed
Final: $t+k.v, s$	

The DRE-ip system records each audited and confirmed vote on the BB in a tabulated form given in Table 2.

The Theorem 1 in [4] analyzes the security of the DRE-ip algorithm. The Theorem 1 states that, in DRE-ip, assuming that all proofs of well-formedness are proofs of knowledge, if the public well-formedness and the tally verification succeed, then the reported tally is the correct tally of all confirmed votes on the bulletin board.

In [4], Shahandashti et al. also consider an intrusive adversary that apart from the ability to determine an arbitrary number of votes, gets read access to the DRE storage for a period during the voting phase. The authors consider that the adversary can control arbitrary number of voters, hence in effect she can cast an arbitrary number of votes. During the access period, the adversary is able to observe the votes cast and read the partial tally t and partial sum s.

The Theorem 2 of the paper [4] analyzes the ballot secrecy in case of an intrusive attack. It states that, in DRE-ip, assuming that all proofs of well-formedness are zero-knowledge, if the DDH assumption holds, then an adversary that determines an arbitrary number of votes and gets temporary read access to the DRE storage cannot get any information about the non-adversarial votes cast before and after the adversarial access period other than their partial tallies.

4.2. Analysis of the protocol

(1) **Weakness.** If the voting machine and the election authority collaborate, the following attack is possible. The election authority who is responsible for publishing the total number of voters cast their votes in the election needs to increase the total count of cast votes. The attacker can add some valid ballots in favor of her chosen candidate at the end of the bulletin board in such a way that it cannot be detected by the DRE-ip tally verification algorithm.

Suppose an attacker would like to add some ballots in favor of her candidate. The attacker can choose some random numbers $r_1, r_2, \ldots, r_k$ from $\mathbb{Z}_q$ such that $\sum_{i=1}^{k} r_i \equiv 0 \pmod{q}$. The attacker generates encrypted ballots of the form $((j+i: U_i, V_i, P_{WF}\{V_i\}), confirmed)$, where $U_i = g_1^{r_i}, V_i = g_2^{r_i} g_2^{v}, P_{WF}\{V_i\} = P_{WF}\{V_i : g_1, g_2, U_i\} = P_K\{r_i : ((U_i = g_1^{r_i}) \wedge (V_i = g_2^{r_i})) \vee ((U_i = g_1^{r_i}) \wedge (V_i/g_2 = g_2^{r_i}))\}$ and v ($v \in \{0,1\}$) is the vote in favor her candidate. Since the attacker knows r_i and v, she can generate the non-interactive zero-knowledge proof $P_{WF}\{V_i\}$.

If an attacker posts encrypted ballots in favor of her candidate in the above form (Table 3) with zero-knowledge proof of well-formedness $P_{WF}\{V_i\}$ and changes the final tally from t to $(t + k.v)$ as soon as the DRE publishes the final tally t on the BB, then such attack cannot be detected by the tally verification procedure of the DRE-ip system. The attacker does not need to change the sum s. This is because $g_1^{\sum_{i=1}^{k} r_i} = g_2^{\sum_{i=1}^{k} r_i} = g_2^0 = e$, the identity element of the group. The attacker can increase the final tally to $(t + k.v)$. The tally verification procedure verifies all zero-knowledge proofs and checks the following equations: $\prod_{j\in\mathbb{C}} U_j = g_1^s, \prod_{j\in\mathbb{C}} V_j = g_2^s . g_2^{t+k.v}$, which will succeed in this case. The first of the two tally verification equations: $\prod_{j\in\mathbb{C}} U_j = g_1^{s+\sum_{i=1}^{k} r_i} = g_1^s g_1^{\sum_{i=1}^{k} r_i} = g_1^s$ and the second of the two tally verification equations: $\prod_{j\in\mathbb{C}} V_j = g_2^{s+\sum_{i=1}^{k} r_i} g_2^{t+k.v} = g_2^s g_2^{\sum_{i=1}^{k} r_i} g_2^{t+k.v} = g_2^s g_2^{t+k.v}$. This is a weakness of the system. The above mentioned ballots can be added at the end of the bulletin board. Note that the proof of well-formedness of the above mentioned confirmed ballots are correct, and hence the tally verification procedure of DRE-ip algorithm [4] satisfies. The DRE-ip algorithm is developed to eliminate requirement of the tallying authority. However, as described above, an attacker can mount the above attack in the following scenario:

(i) The bulletin board is secure and append-only; however, the adversary gets access to the DRE machine to get the signature of above ballots. Additionally, if the election authority colludes with the adversary, an attacker can post the above mentioned ballots at the end of the bulletin board. The signature key can be compromised at the setup stage, or a malicious DRE-ip software can generate such ballots.

Possible countermeasure. To prevent the weakness 1, we assume that the names of voters who have cast their votes are not published on the bulletin board since it has a disadvantage: everybody learns who abstained from voting. Such information in some cases might be illegal to be revealed (see, for example, [70]). We have provided a secure and verifiable voter registration and authentication procedure that will generate a token. This token is presented to the DRE machine. The DRE checks the validity of the token. If the verification succeeds, the DRE allows the voter to cast her vote. We have described this in detail in a later section.

To prevent this kind of attack, in our proposed voting scheme discussed later, we include hash of the previous ballot and a NIZK proof of a valid token number (generated by the voter authentication procedure) with each ballot. This creates an append-only list of ballots that cannot be modified by an adversary. The hash of the last ballot in the election is published with the final tally message along with two NIZK proofs: one of these two NIZK proofs proves the knowledge of the partial sum s; the other NIZK proof proves the knowledge of $s.prev\_hash$.

Note that the idea of a running hash is not new in the e-voting system. In 2007, Sandler and Wallach described the idea of hash linking of votes to ensure the integrity of their system [71] and later applied in the VoteBox system [72]. In 2011, Benaloh and Lazarus proposed a similar idea to mitigate their Trash attack [13], and in 2013, Bell et al. used a similar idea in their Star-Vote system [26]. We extend a similar idea to our proposed e-voting system.

5. Our proposed voter registration and authentication scheme

In this section, we propose a secure and verifiable voter registration and authentication algorithm. In our scheme, a biometric encryption algorithm is used to bind a secret key with her biometric data (fingerprint). This secret key is one of the secret keys used for verifying the authenticity of the voter. We first describe the voter registration algorithm.

Our proposed system requires a fingerprint scanner with fingerprint pulse at the sensor and a smart card reader. The smart card reader must be attached to a device that will verify the eligibility of the voter. It generates a token that will be presented to the DRE machine to proceed with the voting process. It also requires a public bulletin board to display the voter registration data.

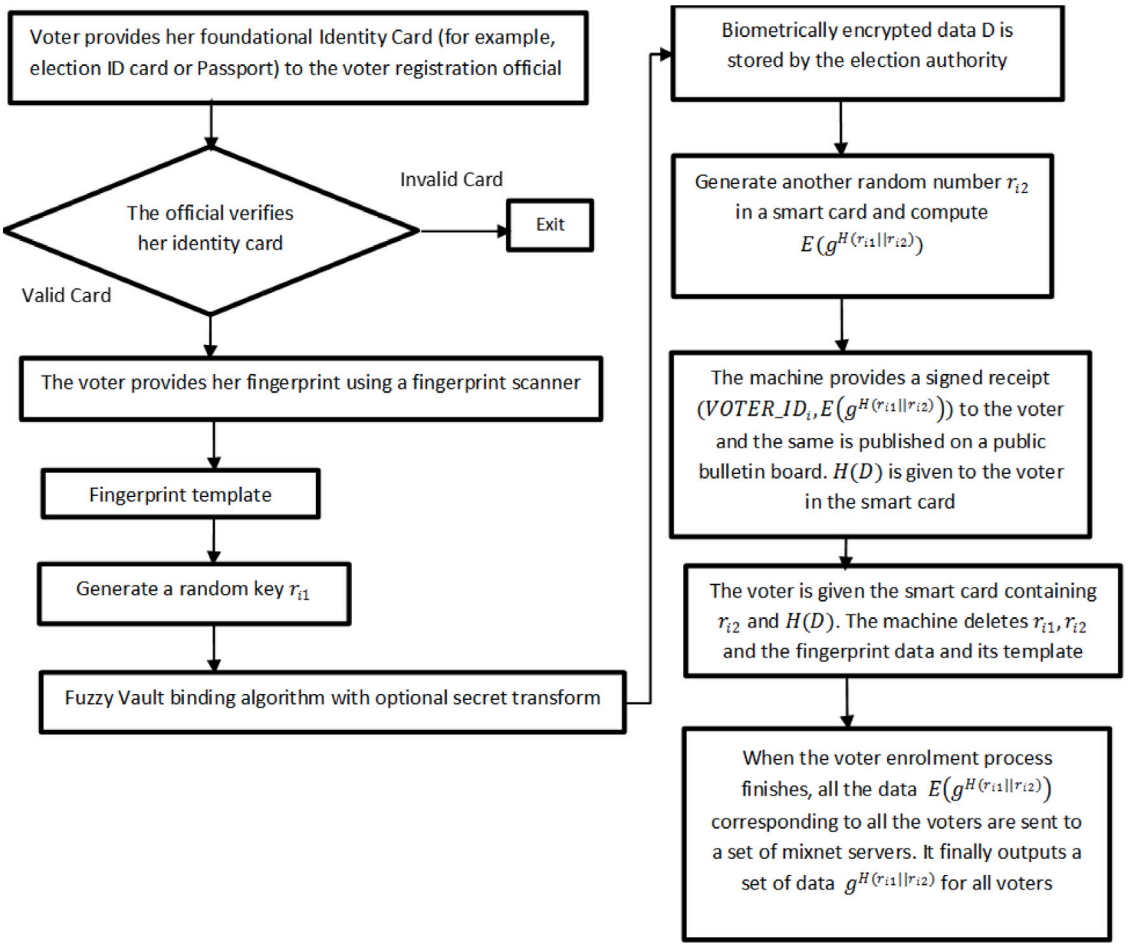

Fig. 1. A logical workflow of our proposed voter registration mechanism.

5.1. Voter registration

In this phase, voters will provide their personal information including voter identification number and fingerprint to register for the election. The voter registration process may occur throughout a year before the election. The voter registration data corresponding to each voter are displayed on a public bulletin board so that the public can verify the voter's eligibility and one-person-one-vote requirement. Fig. 1 depicts a logical workflow of our proposed voter registration mechanism. The following steps are performed to register a voter for the election.

1. The ith voter provides her foundational identity card (for example, election ID card or Passport) to the voter registration official at the registration center. The officer verifies the election ID card. Then the voter has her fingerprint scanned at the registration center.

2. We use a biometric encryption algorithm proposed by Nandakumar et al. [22], called Fuzzy Vault, to bind a randomly generated key into the voter's fingerprint data. As shown in Fig. 2(a), a random number r_{i1} is generated on enrollment uniformly and independently from $\mathbb{Z}_q$ so that neither the voter nor anybody knows it. This random number acts as one of the secret keys used in the registration process. The random number r_{i1} itself is independent of the biometric, and hence it can be changed or modified.

In [22], the secret random key r_{i1} is represented as coefficients of a polynomial in a Galois Field, for example, $GF(2^{16})$. This scheme is designed to secure a key of length $16n$ bits, where n is the degree of the encoding polynomial. For example, if $n = 8$, we can secure a key of size 128 bits. The 16 bit x-coordinate value of the polynomial comprises the fingerprint minutiae location and the angle, and the corresponding Y-coordinate is computed from the polynomial at the point x. Both the values x and y are stored with chaff points to hide the real minutiae. It also stores fingerprint alignment information.

Since Fuzzy Vault actually stores real minutiae even though they are buried inside the chaff points, this may become a source of potential vulnerability [50,52,53]. As a countermeasure, in [50], Jain et al. proposed a transform-in-the-middle approach (shown in the dashed-square in Figs. 2(a) and 2(b)) in which the fingerprint minutiae data are permuted based on the secret password of the user.

3. The machine at the registration center generates another random number r_{i2} uniformly and independently from $\mathbb{Z}_q$. Then it computes $E(g^{H(r_{i1}\|r_{i2})})$, where g is the generator of the group $\mathbb{G}_q$, $E(g^{H(r_{i1}\|r_{i2})})$ is the encryption of $g^{H(r_{i1}\|r_{i2})}$ and '||' is the concatenate operation. We discuss this encryption procedure E later in this section. Then the machine provides a signed receipt consisting $(VOTER\_ID_i, E(g^{H(r_{i1}\|r_{i2})}))$ to the voter.

4. The random number r_{i2} is given to the voter in a smart card or token. The voter keeps the smart card safely with her. It will be required during the voter authentication phase.

5. It publishes $(VOTER\_ID_i, E(g^{H(r_{i1}\|r_{i2})}))$ on a public bulletin board. We assume that the device sends all the data to the BB over an authenticated channel using digital signature. The biometrically encrypted data D_i along with $VOTER\_ID_i$ is securely stored by the election authority in a database or locally (for example, a token or smart card). A hash of this biometrically encrypted data D_i, $H(D_i)$, is also provided to the voter in her smart card (that also contains the random number r_{i2}). The biometrically encrypted data D_i will be used for verification of the voter during the voter authentication phase.

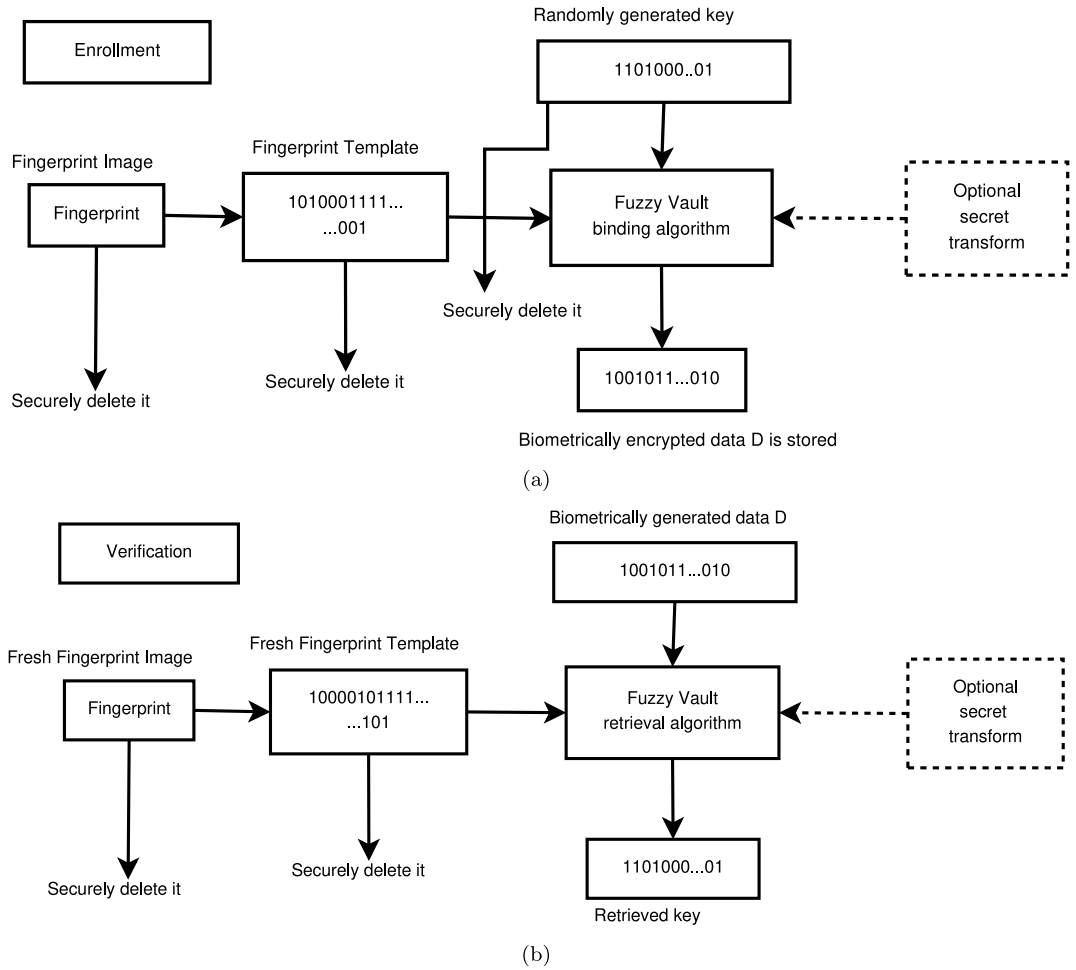

Fig. 2. High level diagram of the Fuzzy Vault process to bind a key (r_{i1} in the description) with the fingerprint. (a) Enrollment process that binds a randomly generated key (r_{i1} in the description); (b) Verification process that retrieves the same key (r_{i1} in the description).

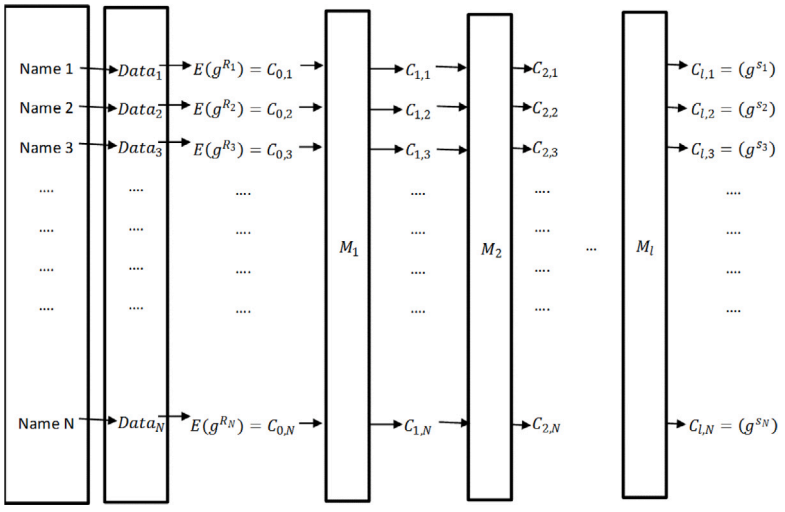

Fig. 3. Public bulletin board displaying voter registration data of N voters along with l mixnet servers. Here, Name i represents the name of the ith voter. $Data_i$ represents the ith voter's data $(VOTER\_ID_i, E(g^{H(r_{i1}\|r_{i2})}))$. R_i represents $H(r_{i1}\|r_{i2})$. M_k are mixnet servers $\forall k \in \{1, 2, \ldots, l\}$. $C_{k-1,j}, \forall j \in \{1, 2, \ldots, N\}$ are the inputs to the mixnet M_k, and its outputs are $C_{k,j}, \forall j \in \{1, 2, \ldots, N\}$. The last mixnet server's output $(g^{s_1}, g^{s_2}, \ldots, g^{s_N})$ is a permutation of $(g^{R_1}, g^{R_2}, \ldots, g^{R_N})$.

It securely deletes the random numbers r_{i1}, r_{i2} and the fingerprint template.

6. The voter leaves the registration center and checks that her receipt is recorded on the public bulletin board.

7. When the enrollment process finishes, all the receipts $(VOTER\_ID_i, E(g^{H(r_{i1}\|r_{i2})}))$ are published on the public bulletin board. We use some mixnet servers proposed by Neff et al. [64] to permute the set of data $g^{H(r_{i1}\|r_{i2})}$ and publish it on the bulletin board. The input to

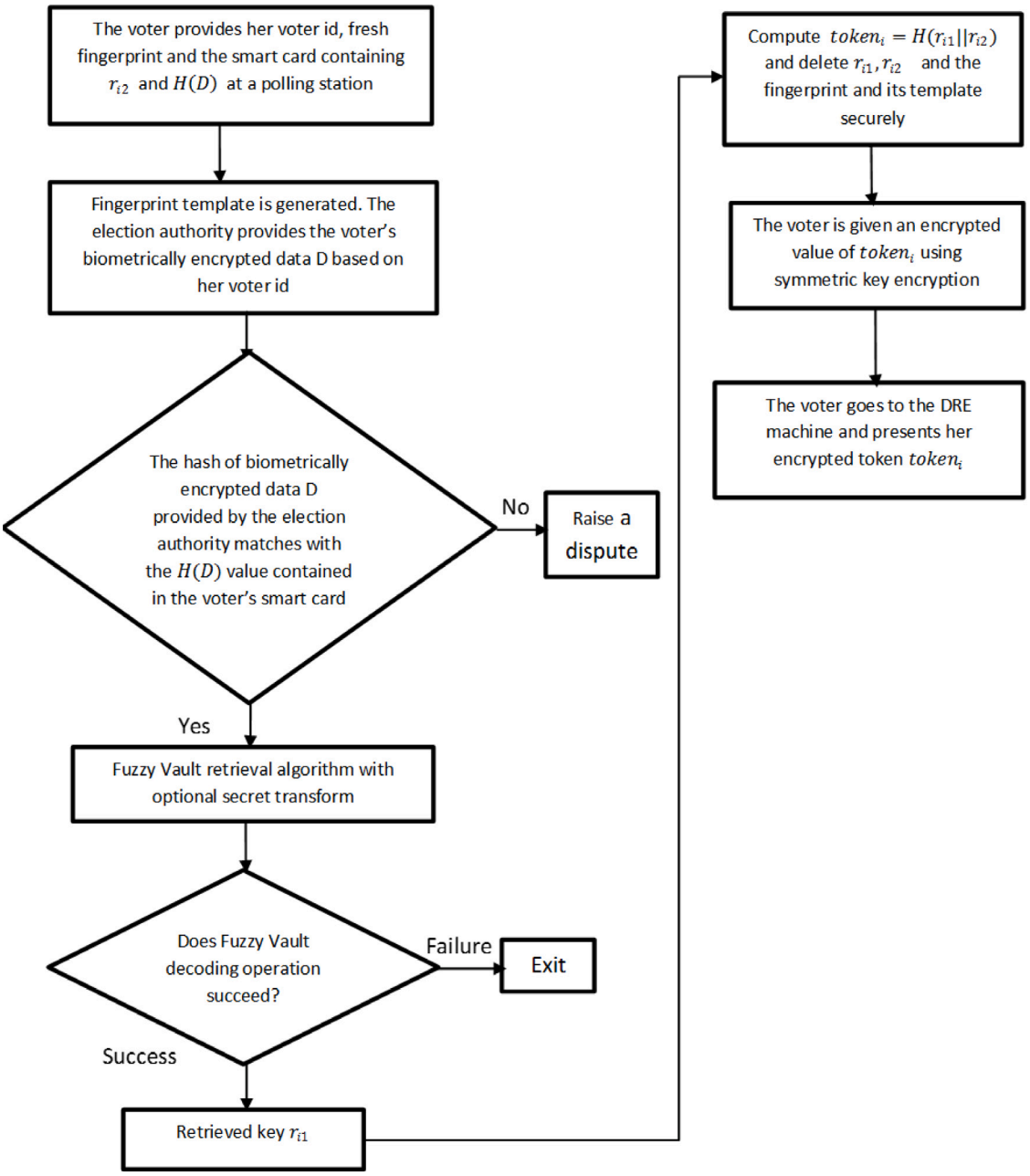

Fig. 4. A logical workflow of our proposed voter authentication mechanism.

the mixnet are $(E(g^{H(r_{i1}\|r_{i2})})), \forall i \in \{1, 2, \ldots, N\}$, where the encryption algorithm E depends on the mixnet servers [64], N is the total number of voters. It outputs $g^{H(r_{i1}\|r_{i2})}, \forall i \in \{1, 2, \ldots, N\}$ in a verifiable manner. Fig. 3 depicts the mixnet servers. The mixnet [64] servers provide NIZK proofs to prove a permutation between the input data $(E(g^{H(r_{i1}\|r_{i2})})), \forall i \in \{1, 2, \ldots, N\}$ and output $g^{H(r_{i1}\|r_{i2})}, \forall i \in \{1, 2, \ldots, N\}$ without revealing the exact permutation. We assume that at least one mixnet server is honest, who does not reveal its secret keys and the permutation.

5.2. Voter authentication during the voting phase

In this section, we discuss the voter authentication procedure using the biometrically encrypted data collected during the voter registration phase. The voter authentication is performed in parallel with the voting phase. After a voter's verification is successful, it generates a token number that will be used to cast her vote. All the voter's biometrically encrypted data D_i along with $VOTER\_ID_i$ that were collected during the voter registration phase are presented (for example, in a smart card) here at this phase for verification of voters. Fig. 4 shows a logical workflow diagram of our proposed voter authentication mechanism. The following steps are performed to verify the authenticity of a voter.

1. The voter goes to the polling station and provides her $VOTER\_ID_i$, fingerprint, the smart card containing r_{i2} and $H(D_i)$ to verify her eligibility to vote. Note that the hash of the biometrically encrypted data D_i provided by the election authority should match with $H(D_i)$ value stored in the voter's smart card; otherwise, the voter should raise a dispute.

2. We use the verification procedure of the Fuzzy Vault algorithm proposed by Nandakumar et al. [22] to verify her eligibility to vote. Fig. 2(b) illustrates the verification procedure of the Fuzzy Vault algorithm [22]. On verification, the user provides her fresh fingerprint data which, when applied to the legitimate biomerically encrypted data D_i, will let the Fuzzy Vault algorithm recreate the same key r_{i1}. If

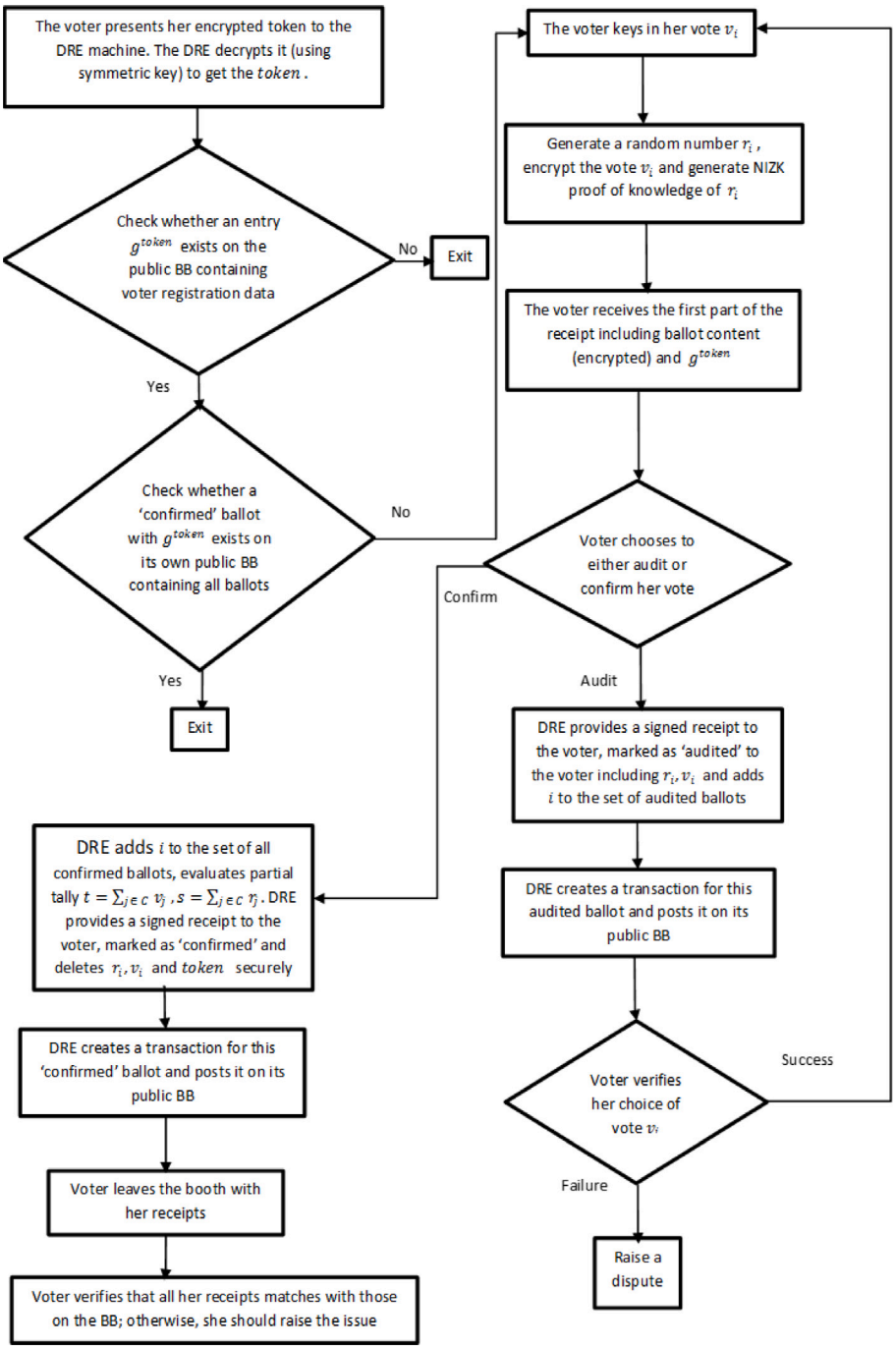

Fig. 5. A logical workflow of our proposed voting procedure.

a sufficient number of minutiae points coincide with some genuine recorded points, the full polynomial can be constructed using an error-correcting code (for example, Reed–Solomon error-correcting code) or Lagrange interpolation. The secret will be correctly decrypted only if the polynomial is correctly constructed. The performance results show that FRR (False Rejection Rate) is about 6% to 17% and FAR (False Acceptance Rate) is about 0.02%. The FRR is due to poor samples that are fact of life in biometric systems. However, the FRR could be significantly reduced through retries. The Fuzzy Vault algorithm is designed to account for an acceptable variations of the input fingerprint. On the other hand, an imposter with different enough fingerprint will not be able to recreate the key.

3. It provides $token_i = H(r_{i1} \| r_{i2})$ as a token number to the voter. The token number $token_i$ is given to the voter in an encrypted format using a symmetric key encryption. At the end of verification procedure, it securely deletes $r_{i1}, r_{i2}, token_i$, the fingerprint and the fingerprint template once again.

4. The voter goes to the DRE machine and presents her encrypted token.

Voters must observe the public bulletin board (Fig. 3) throughout the election stage to ensure that their registration data are not being changed. If, at any point between the voter registration and the final tallying phase, any of the entry on the public bulletin board is changed based on her receipt, the voter must raise an issue with the election official. Note that the proposed voter registration and authentication scheme can also be incorporated into other verifiable e-voting systems.

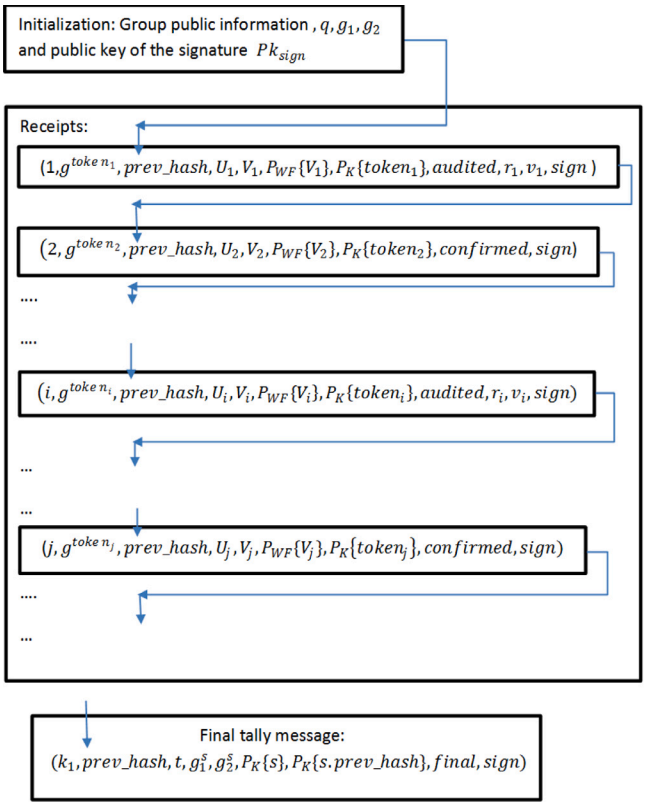

Fig. 6. A public bulletin board of our system.

5.3. Performance analysis of the voter registration and authentication scheme

The voter registration phase is conducted well before the actual election. For each voter, the voter registration procedure includes execution of the Fuzzy Vault algorithm and computation of one encryption $E(g^{H(r_{i1}\|r_{i2})})$, where the encryption algorithm E depends on the mixnet server [64]. The mixnet [64] requires $8N$ exponentiations, where N is the total number of voters registered for the election (in an electoral constituency). Therefore, the computational time complexity of the voter registration scheme involves execution of the Fuzzy Vault algorithm and $O(N)$ exponentiations for Mixnet. The voter registration may be done throughout a year before the election. Voter authentication includes execution of the Fuzzy Vault key retrieval algorithm and computation of a symmetric key encryption of $H(r_{i1}\|r_{i2})$.

6. Our proposed voting system

In this section, we propose a voting scheme, analyze its security properties, propose efficient NIZK proofs, compare it with some well-known DRE-based voting systems and finally measure its performance. It requires a DRE machine with a printer attached to it and a public bulletin board to show the recorded ballots in public. The bulletin board can be a publicly accessible web site.

6.1. Proposed voting scheme

The voting scheme is described in this section.

6.1.1. Voting and tallying phase

We now describe the algorithm for the voting phase and the tallying phase. The DRE-ip system is modified to prevent weakness 1 and the ballot stuffing attack. In our system, the BB can be insecure. However, if the adversary tries to modify the content of the BB, it could be detected by the public because of the use of hashchain. Let us assume that one DRE machine is used to elect a candidate in a regional zone. Extension to multiple DRE machines has been discussed in a later section. We assume that the DRE sends recorded ballots to BB over an authenticated channel using standard technique such as digital signatures. For simplicity, the algorithm is described for the case where there are only two candidates i.e. if v_i represents the vote for the ith ballot, we have $v_i \in \{0, 1\}$. Extension to multiple candidates has been discussed in a later section. An audited ballot is not used to cast a vote. Let $\mathbb{A}$, $\mathbb{C}$ and $\mathbb{B}$ are the set of all audited ballots, confirmed ballots and all ballots respectively. Thus, at the end of the voting phase, $\mathbb{B} = \mathbb{A} \cup \mathbb{C}$. We use the exponential ElGamal cryptosystem to encrypt the votes in which no party knows the decryption key. Fig. 6 depicts a public bulletin board of our proposed e-voting system.

Key Generation Phase. 1. The system generates an efficient description of a cyclic group $\mathbb{G}_q$ of order q with two distinct generators g_1, g_2 whose logarithmic relationship is unknown.

2. The DRE publishes description of the group $(\mathbb{G}_q, q, g_1, g_2, Pk_{sign})$ on a public BB, where Pk_{sign} is the public key of the digital signature scheme (for example, DSA) used by the DRE machine. Initially, $t = 0$ and $s = 0$. The DRE calculates hash of this transaction and stores it in $prev\_hash$. The hash value $prev\_hash$ will be included in the next ballot. This hash must be verified by public during the tallying phase to ensure that this transaction remains unaltered.

Voting Phase. This phase involves the voter, the DRE and the BB. Fig. 5 illustrates a logical workflow diagram of our proposed voting procedure.

1. The voter goes to the DRE machine and presents her encrypted token. The DRE decrypts (using symmetric key) it to get the $token$ (generated during the voter authentication phase). The DRE checks for an entry g^{token} on the voter registration bulletin board (Fig. 3) corresponding to her token. It also verifies that this token has not been already used to cast a vote, by checking all confirmed ballots on its own public bulletin board (that displays all ballots, Fig. 6). If the verification succeeds, it allows the voter to proceed to the next step.

2. The voter initiates the voting and keys in her vote $v_i \in \{0, 1\}$.

3. The DRE generates random $r_i \in \mathbb{Z}_q^*$, and evaluates

$$U_i = g_1^{r_i}, V_i = g_2^{r_i} g_2^{v_i},$$
$$P_{WF}\{V_i : g_1, g_2, U_i\}$$
$$= P_K\{r_i : ((U_i = g_1^{r_i}) \wedge (V_i = g_2^{r_i})) \vee ((U_i = g_1^{r_i}) \wedge (V_i/g_2 = g_2^{r_i}))\}$$
$$= P_K\{r_i : (U_i = g_1^{r_i}) \wedge ((V_i = g_2^{r_i}) \vee (V_i/g_2 = g_2^{r_i}))\},$$

and $P_K\{token : (W_i = g^{token})\}$.

Here, $P_{WF}\{V_i : g_1, g_2, U_i\}$ is a NIZK proof to show that the ballot is an encryption of either $v_i = 0$ or $v_i = 1$ (i.e. the ballot is well-formed). $P_K\{r_i\}$ is a NIZK proof of knowledge of r_i. Algorithm 3 (resp. Algorithm 4) and Algorithm 5 (resp. Algorithm 6) describe the creation (resp. verification) of the NIZK proof $P_{WF}\{V_i\}$ and $P_K\{token\}$ respectively. The DRE machine provides a signed receipt including the unique ballot index i and the ballot content $(i, g^{token}, prev\_hash, U_i, V_i, P_{WF}\{V_i\}, P_K\{token\}, sign)$ to the voter, where $sign$ is the signature of $(i, g^{token}, prev\_hash, U_i, V_i, P_{WF}\{V_i\}, P_K\{token\})$.

4. The voter receives the first part of the receipt and chooses to either audit or confirm her vote.

In case of audit:

5. The DRE adds i to $\mathbb{A}$. The DRE provides a signed receipt of the audit, marked as 'audited', including r_i, v_i to the voter.

6. The DRE merges both parts into a single part $((i, g^{token}, prev\_hash, U_i, V_i, P_{WF}\{V_i\}, P_K\{token\}), audited, r_i, v_i, sign)$, where $sign$ is the signature of $((i, g^{token}, prev\_hash, U_i, V_i, P_{WF}\{V_i\}, P_K\{token\}), audited, r_i, v_i)$. The DRE posts this receipt (or transaction) on the BB. The DRE computes hash of this transaction and updates $prev\_hash$ value. This $prev\_hash$ will be attached to the next ballot (or transaction).

7. The voter takes and keeps the receipt. She verifies her vote v_i. If the verification succeeds, it continues to execute from step 2; otherwise, the voter should raise a dispute.

In case of confirmation:

5. The DRE adds i to the set $\mathbb{C}$ and updates the tally and the sum:

$$t = \sum_{j\in\mathbb{C}} v_j, s = \sum_{j\in\mathbb{C}} r_j. \quad (1)$$

The DRE provides a signed receipt, marked as 'confirmed' to the voter. Then the DRE securely deletes r_i, v_i and $token$.

6. The DRE merges both parts into a single part, $((i, g^{token}, prev\_hash, U_i, V_i, P_{WF}\{V_i\}, P_K\{token\}), confirmed, sign)$, where $sign$ is the signature of $((i, g^{token}, prev\_hash, U_i, V_i, P_{WF}\{V_i\}, P_K\{token\}), confirmed)$. The DRE posts this transaction on the BB. The DRE computes hash of this transaction and stores it in $prev\_hash$. This $prev\_hash$ will be attached to the next ballot (or transaction).

7. The voter leaves the polling booth with her receipts.

8. The voter verifies that all her receipts match with those on the BB. The voter should raise an issue if her receipts do not match with those on the BB.

Verification by bulletin board (cloud or blockchain). This phase involves the DRE and the underlying BB. The BB can be implemented using cloud server, InterPlanetary File System (IPFS), Ethereum blockchain (method 1) or a combination of both blockchain and cloud server/IPFS (method 2). The BB must verify the signature of the ballot before storing it. The BB may optionally verify the ballot well-formedness proof included in the ballot before adding it into the BB.

Tallying Phase. This phase involves the DRE, the BB and the public.

1. The DRE evaluates: $\Gamma_1 = g_1^s$, $\Gamma_2 = g_2^s$, $\Gamma_3 = g_1^{s.prev\_hash}$ and $\Gamma_4 = g_2^{s.prev\_hash}$. Then the DRE posts the final tally t, Γ_1, and Γ_2 on the BB with zero-knowledge proofs $P_K\{s : (\Gamma_1 = g_1^s) \wedge (\Gamma_2 = g_2^s)\}$ and $P_K\{s.prev\_hash : (\Gamma_3 = g_1^{s.prev\_hash}) \wedge (\Gamma_4 = g_2^{s.prev\_hash})\}$. The Algorithm 7 (resp. Algorithm 8) provided in Appendix A describes the procedure for generation (resp. verification) of these NIZK proofs. Let the tuple $(t, \Gamma_1, \Gamma_2, P_K\{s\}, P_K\{s.prev\_hash\})$ be denoted by 'MESSAGE'. To post ith message, say 'MESSAGE', on the BB, the following procedure is adopted.

(i) The DRE creates a transaction consisting of the data $(i, prev\_hash, MESSAGE, final, sign)$, where $sign$ is the signature of $(i, prev\_hash, MESSAGE, final)$. The DRE posts this transaction on the BB.

2. The public:

(i) verify that the hash of each transaction matches with the $prev\_hash$ value of the next ballot (or transaction).

(ii) verify that the zero-knowledge proofs provided in step 1 of this tally phase are correct, provided the hash of the last ballot on the bulletin board. This step involves verification of the two NIZK proofs: (a) verification of $P_K\{s\}$, given Γ_1 and Γ_2; (b) verification of $P_K\{s.prev\_hash\}$, provided Γ_3 ($= \Gamma_1^{prev\_hash}$) and Γ_4 ($= \Gamma_2^{prev\_hash}$), where $prev\_hash$ is the hash of the last ballot on the bulletin board.

(iii) verify that all the well-formedness proofs of each transaction on the BB (well-formedness verification) are correct.

(iv) verify that for all the audited ballots $((i, g^{token}, prev\_hash, U_i, V_i, P_{WF}\{V_i\}, P_K\{token\}), audited, r_i, v_i, sign)$ on the BB: the first part of the receipt $(i, g^{token}, prev\_hash, U_i, V_i, P_{WF}\{V_i\}, P_K\{token\})$ is consistent with r_i and v_i.

(v) verify that the signature of each transaction is valid.

(vi) verify that all the NIZK proofs provided by the mixnet servers (Fig. 3 in Section 5.1) are correct.

(vii) verify that all g^{token} in all confirmed ballots on the BB (Fig. 6) are different and there exists only one entry with value g^{token} on the BB generated during the voter registration phase (Fig. 3).

(viii) verify that all the NIZK proofs $P_K\{token\}$ are correct.

(ix) verify that all the following equations hold (tally verification):

$$\prod_{j\in\mathbb{C}} U_j = \Gamma_1, \text{ and } \prod_{j\in\mathbb{C}} V_j = \Gamma_2 . g_2^t. \quad (2)$$

During the voting and tallying phase, if any of the verification carried out by the voter or the public fails, an issue should be raised with the election authority. We assume that there are procedures in place to deal with such verification failures. In practice, a truncated hash function may be used to calculate short digest e.g. 32 bit long for each part of the receipt so that a voter can easily compare their receipt with those on the bulletin board. In this case, voters are expected to verify their receipts with those on the bulletin board. We assume that there are facilities provided for them to do so in the polling station.

If sufficient resources are available, there can be another optional module that takes a transaction from the proposed algorithm and checks whether it has been added to the BB.

6.1.2. Extension to multiple DRE machines

If multiple DRE machines are used in a regional zone (or electoral constituency) to elect a candidate, then instead of disclosing the tally from each DRE machine and then adding them together to get the final tally, we publish the final tally by combining the tallies from all DRE machines so that the tally from each DRE machine remains secret. The correctness of the procedure are realized by producing the corresponding zero-knowledge proof. This is particularly done to avoid revealing the distribution of voters against various political parties in small areas (where DRE machines are used to cast the vote) of a regional zone. Let the total number of DRE machines used in a regional zone be $\eta (\eta \in \mathbb{N})$. Let $t_{(i)}$ and $s_{(i)}$ represent the tally t and sum of random variables s respectively for the ith DRE machine.

1. Each DRE performs following tasks.

(i) The ith DRE posts $\Gamma_{2(i)} = g_2^{t_{(i)}+s_{(i)}}$ on the BB instead of total tally $t_{(i)}$ with $P_K\{t_{(i)} + s_{(i)} : \Gamma_{2(i)} = g_2^{t_{(i)}+s_{(i)}}\}$, the non-interactive zero-knowledge proof of knowledge of the sum of tally $t_{(i)}$ and $s_{(i)}$ for this DRE machine. The DRE also posts $\Gamma_{1(i)} = g_1^{s_{(i)}}$ with $P_K\{s_{(i)} : \Gamma_{1(i)} = g_1^{s_{(i)}}\}$, the zero-knowledge proof of the sum of random variables $s_{(i)}$ for all confirmed ballots.

(ii) Provided above information, the public perform all tasks stated in step 2 of the *Tallying Phase*.

2. (a) Let the group information g_1 and g_2 be same for every DRE machine. If they are different for each DRE machine, we perform step (b) instead of this step.

(i) Now all DRE machines perform two secure multi-party computations, proposed later in this section, to evaluate $t_{final} = \Sigma_{i=1}^{\eta} t_{(i)}$ and $s_{final} = \Sigma_{i=1}^{\eta} s_{(i)}$. It publishes t_{final} and s_{final} as the final tally and the final sum of random variables.

(ii) The public verify that $\prod_{i=1}^{\eta} \Gamma_{2(i)} = g_2^{t_{final}} g_2^{s_{final}}$ and $\prod_{i=1}^{\eta} \Gamma_{1(i)} = g_1^{s_{final}}$.

(b) Let $g_{1(i)}$ and $g_{2(i)}$ represent the group information g_1 and g_2 respectively for the ith DRE machine.

(i) Since, for every ith DRE, $g_{1(i)}$ is public information, each DRE can send their $g_{1(i)}$ to each other to compute $g_a = \prod_{i=1}^{\eta} g_{1(i)}$ and $g_b = \prod_{i=1}^{\eta} g_{2(i)}$. We assume that DRE machines can communicate with each other over an authenticated channel using digital signatures. It evaluates: $\Gamma_{b(i)} = g_b^{t_{(i)}+s_{(i)}}$, $\Gamma_{2(i)} = g_{2(i)}^{t_{(i)}+s_{(i)}}$, $\Gamma_{a(i)} = g_a^{s_{(i)}}$, $\Gamma_{1(i)} = g_{1(i)}^{s_{(i)}}$.

(ii) Each DRE posts $(\Gamma_{b(i)}, \Gamma_{2(i)}), (\Gamma_{a(i)}, \Gamma_{1(i)})$ on the BB with $P_K\{t_{(i)} + s_{(i)} : \Gamma_{b(i)} = g_b^{t_{(i)}+s_{(i)}} \wedge \Gamma_{2(i)} = g_{2(i)}^{t_{(i)}+s_{(i)}}\}$ and $P_K\{s_{(i)} : \Gamma_{a(i)} = g_a^{s_{(i)}} \wedge \Gamma_{1(i)} = g_{1(i)}^{s_{(i)}}\}$. The public must verify these proofs.

(iii) Now all DRE machines perform two secure multi-party computations, proposed later in this section, to evaluate $t_{final} = \Sigma_{i=1}^{\eta} t_{(i)}$ and $s_{final} = \Sigma_{i=1}^{\eta} s_{(i)}$. It publishes t_{final} and s_{final} as the final tally and the final sum of random variables respectively.

(iv) The public verify that $\prod_{i=1}^{\eta} \Gamma_{b(i)} = g_b^{t_{final}} g_b^{s_{final}}$ and $\prod_{i=1}^{\eta} \Gamma_{a(i)} = g_a^{s_{final}}$.

Secure multi-party computations to compute Σt. Now we briefly describe a secure multi-party computation protocol to compute the sum of various parties' private inputs. Suppose three parties have private inputs t_1, t_2, t_3 respectively. Our secure multi-party computation protocol runs in two rounds which is also the minimum number of rounds

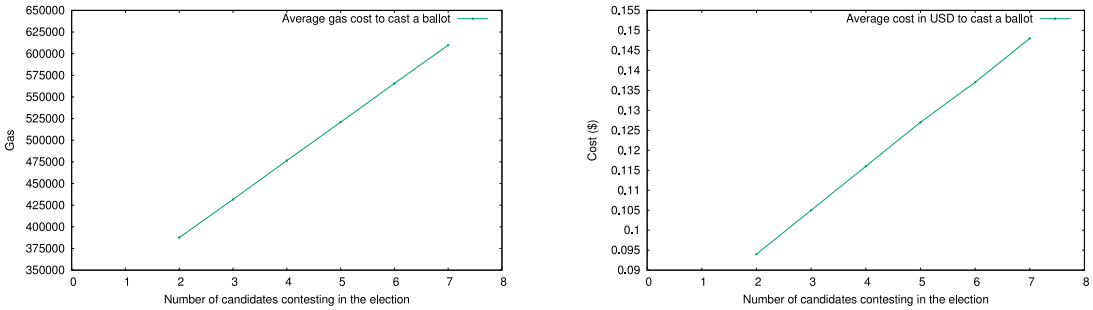

Fig. 7. (a) Gas cost for casting a ballot based on the number of candidates contesting the election. (b) Costs for casting a ballot based on the number of candidates contesting the election. The costs are approximated in USD ($) using the conversion rate of 1 Ether = $243 and the gas price of 0.000000001 ether that are real world costs in July, 2020. Here, for both (a) and (b), casting a ballot involves storing a ballot on the blockchain without verifying the ballot well-formedness proof (NIZK proof).

required to compute a function securely in any secure multi-party computation [73].

In the first round, the first party chooses random $t_{11} \in_R \mathbb{Z}_q$, and $t_{12} \in_R \mathbb{Z}_q$ uniformly and independently. Then it computes $t_{13} = (t_1 - t_{11} - t_{12})$. The first party sends t_{12} to the second party and t_{13} to the third party in an encrypted format over an authenticated channel. Similarly, the second party chooses random $t_{21} \in_R \mathbb{Z}_q$, and $t_{22} \in \mathbb{Z}_q$ uniformly and independently. Then it computes $t_{23} = (t_2 - t_{21} - t_{22})$. The second party sends t_{21} to the first party and t_{23} to the third party in an encrypted format over an authenticated channel. Similarly, the third party chooses random $t_{31} \in_R \mathbb{Z}_q$, and $t_{32} \in \mathbb{Z}_q$ uniformly and independently. Then it computes $t_{33} = (t_3 - t_{31} - t_{32})$. The third party sends t_{31} to the first party and t_{32} to the second party in an encrypted format over an authenticated channel. Now, after receiving t_{21} and t_{31} from the second and the third party respectively, the first party computes $T_1 = t_{11} + t_{21} + t_{31}$. Similarly, the second party computes $T_2 = t_{12} + t_{22} + t_{32}$ and the third party computes $T_3 = t_{13} + t_{23} + t_{33}$.

In the second round, all the three parties send their T_i's to a fourth party who computes $T = T_1 + T_2 + T_3$, which is the sum of t_1, t_2, t_3.

Note that the private input t_i of each party remains secret after computation of their sum for all $i \in \{1, 2, 3\}$. In a similar fashion, untrusted parties can compute the sum of their private inputs without revealing it. The zero-knowledge proofs, verification by the public and the final tally verification checks (step 2 of the *Tallying Phase*) collectively prove that all parties follow the protocol faithfully.

6.1.3. Extension to multiple candidates

If there are $n(n >= 3)$ candidates contesting the election, we will consider an upper bound, say M, on the number of voters and will encode the vote for the jth candidate as $v = M^{j-1}$. The ith ballot in that case will of the form $((i, g^{token}, prev\_hash, U_i, V_i, P_{WF}\{V_i\}, P_K\{token\}), audited, r_i, v_i, sign)$ in case of audit or $((i, g^{token}, prev\_hash, U_i, V_i, P_{WF}\{V_i\}, P_K\{token\}), confirmed, sign)$ in case of confirmed vote, where $V_i = g_2^{r_i} g_2^{M^{j-1}}$. The well-formedness proof $P_{WF}\{V_i\}$ will be 1-out-of-n disjunctive proof and can be stated as:

$$P_{WF}\{V_i : g_1, g_2, U_i\} = P_K\{r_i : \vee_{j=1}^{n}((U_i = g_1^{r_i}) \wedge (V_i/g_2^{M^{j-1}} = g_2^{r_i}))\}$$
$$= P_K\{r_i : (U_i = g_1^{r_i}) \wedge (\vee_{j=1}^{n}(V_i/g_2^{M^{j-1}} = g_2^{r_i}))\}.$$

The well-formedness proof $P_{WF}\{V_i\}$ is a NIZK proof to show that the ballot is an encryption of v_i, where $v_i \in \{1, M, M^2, \ldots, M^{n-1}\}$.

6.2. Storing recorded ballots

Depending on how the election is arranged, there can be several methods to store the ballots. In our modified DRE-ip system, any ballot is linked to the previous ballot by using the hash of the previous ballot in the digital signature of the ballot. This creates an append-only list that can be maintained by anyone. In this section, we highlight two existing methods and propose two new methods to store these ballots.

Using cloud server. The cryptographic group information, the public keys of the ElGamal encryption and digital signature, all the ballots and the final tally messages can be stored on cloud. The public bulletin board accesses the cloud storage to show the ballots. Due to the use of hashchain, any modification on the BB can be detected by the public during the tallying phase. Multiple cloud servers could be used to remove a single point of failure (see, for example, the public BB proposed by Palngipang et al. [74]).

Using InterPlanetary File System (IPFS). All the cryptographic keys, ballots and the final tally messages can be stored in IPFS [75] system which is a distribute file system. The IPFS hashes can be stored separately on a public bulletin board. This will reduce the storage burden on the election authority, but will not eliminate it completely as the election authority needs to store the IPFS hashes anyway for others to be able to download the ballots and other information from the IPFS system.

Using public blockchain (method 1). The cryptographic group information, the public keys of ElGamal encryption and digital signature, all the ballots and the final tally messages can be stored on a public blockchain such as Ethereum. During the setup phase, the DRE sends a transaction containing the description of the group of the form $(\mathbb{G}_q, q, g_1, g_2, Pk_{sign})$ to the blockchain, where Pk_{sign} is the public key of the digital signature scheme. When a voter audits her vote, the DRE sends a transaction containing an audited ballot of the form $((i, g^{token}, prev\_hash, U_i, V_i, P_{WF}\{V_i\}, P_K\{token\}), audited, r_i, v_i, sign)$. When she confirms her vote, the DRE sends a transaction containing a confirmed ballot of the form $((i, g^{token}, prev\_hash, U_i, V_i, P_{WF}\{V_i\}, P_K\{token\}), confirmed, sign)$. Finally, at the tallying phase, the DRE sends a transaction containing the final tally message of the form $(i, prev\_hash, t, \Gamma_1, \Gamma_2, P_K\{s\}, P_K\{s.prev\_hash\}, final, sign)$. The public and the individual voters can verify that their ballots are included into the blockchain. Once a ballot is posted on the blockchain, it remains tamper-proof. The blockchain stores the ballots after verifying its signature. The blockchain may also optionally verify the ballot well-formedness proof (NIZK proof) for each ballot. However, it is not necessary for the blockchain to verify the ballot well-formedness proof (NIZK proof) since it will be verified by the public during the tally phase by a separate module. We have improved the performance of the 1-out-of-n NIZK proof by reducing the number of exponentiations to almost half as compared to the NIZK proof described in [4]. A small cartel of miners (< 51%) may delay transactions from being accepted into the blockchain by using selfish mining or feather forking. Such ability of miners to delay a transaction is a fundamental problem for every smart contract. We discuss the performance analysis (both cost and timing

measurements analysis) of the scheme in detail in a later section. The advantage of this method is that all the cryptographic keys, ballots and final tally messages remain tamper-resistant and can be verified by the public. The disadvantage is that this method may cost more than other methods described in this section. Therefore, it is preferable that the blockchain stores the ballots verifying only its signature, but without verifying any NIZK proof (Figs. 7(a) and 7(b)), since it will reduce the mining cost and all the NIZK proofs will be verified by a separate module in the tallying phase.

Using both the cloud server/IPFS and the public blockchain (method 2). Another method to implement a public bulletin board is to use both the blockchain and the cloud server. We keep the bulk of the data on cloud and a minimal set of critical information on the blockchain so that no one can tamper with the data stored on the cloud without changing the more secure data stored on the blockchain. The cryptographic group information and the public keys of ElGamal encryption and digital signature (i.e. a transaction containing the description of the group of the form $(\mathbb{G}_q, q, g_1, g_2, Pk_{sign})$) are stored on a blockchain such as Ethereum in a single transaction before beginning of the polling phase. According to our proposed protocol, the hash of this transaction is included in the first ballot as well as in the signature of the first ballot. This transaction is the first transaction of the hash chain of ballots. During the tallying phase, the final tally message (i.e. a transaction containing the final tally message of the form $(i, prev\_hash, t, \Gamma_1, \Gamma_2, P_K\{s\}, P_K\{s.prev\_hash\}, final, sign)$) is also stored on the public blockchain. The DRE sends the final tally message to the blockchain and verifies that it has been included into the blockchain. According to our proposed protocol, the hash of the last valid cast ballot is included in the final tally message as well as in the signature of the final tally message. However, all the audited and confirmed ballots during the voting phase are stored on cloud. Thus, if any of the already recorded ballot on the cloud is modified, it can be detected by the public by observing the final tally message transaction and the group information transaction stored in the blockchain. It also prevents a malicious bulletin board from successfully adding a new ballot or deleting a recorded ballot on the cloud without detection. Since, for each DRE machine, we add only two transactions (the first and the last transaction of the hashchain) into the blockchain, the total financial cost towards mining these ballots is at most 16 million gas. Therefore, in this case, the cost of mining is almost negligible. We discuss the timing measurements analysis for posting each ballot on the BB in a later section. A dynamic cloud network control protocol [76] or a queueing model [77] could be used for the scalability and allocation of cloud resources in presence of the dynamic workload of voting citizens. Note that we can use IPFS as an alternative to cloud. However, in that case, we have to store the IPFS hashes on the blockchain in addition to the cryptographic group information and the final tally message. The advantages of this method (method 2) are: (1) the cost for storing the ballots is low, and hence this method is more suitable than method 1 when the number of voters in the election is very large; (2) the cryptographic keys and the final tally messages remain tamper-resistant; (3) all the ballots are tamper-evident. The disadvantage is that all the ballots are tamper-evident, but less tamper-resistant than the ballots while using method 1 (i.e. using blockchain). Note that, as discussed before, irrespective of these two methods (method 1 and method 2), if any ballot is tampered by the adversary, it will be detected by the public in the tally verification phase.

6.3. Security analysis of the proposed system

In this section, we discuss the security properties of the proposed voter registration, voter authentication and the voting scheme. In particular, we prove that our scheme satisfies all the properties described in Section 3.2 as well as the integrity and vote secrecy requirements stated in Section 3.1.

6.3.1. Eligibility verifiability and the security analysis of the voter registration and authentication procedure

In this section, we prove the eligibility verifiability property and the correctness of the voter authentication procedure.

Legitimacy of each voter. On successful verification of a voter, the voter authentication procedure proves three things: the entry corresponds to the fingerprint of the legitimate voter, it can retrieve the correct random number r_{i1} from her biometrically encrypted data D_i and the voter has presented the correct random number r_{i2} by providing her smart card. Therefore, the voter has provided all her secret keys and her fingerprint correctly. An imposter has to know two secret keys r_{i1} and r_{i2} to generate a token number $H(r_{i1}\|r_{i2})$ correctly. To form a valid ballot, the imposter must include $g^{H(r_{i1}\|r_{i2})}$ and a NIZK proof of knowledge of the secret $H(r_{i1}\|r_{i2})$ in the ballot. The public must verify two things: an entry with value $g^{H(r_{i1}\|r_{i2})}$ exists on the output of the mixnet server (i.e. the public bulletin board, Fig. 3) published after the voter registration phase; and the NIZK proof of the secret $H(r_{i1}\|r_{i2})$ provided in the ballot is correct. The random number r_{i1} is biometrically encrypted with the voter's fingerprint. The biometrically encrypted data D_i is kept securely by the election authority. The random number r_{i2} is kept by the voter in a smart card. Therefore, for an imposter, the probability to generate a correct token is $(1/2)^{\kappa}$, where $\kappa = min\{l_4, l_1 + l_2\}$, l_4 is the number of bits in the output of the hash function H, l_1, l_2 are the number of bits in r_{i1} and r_{i2} respectively. During the voter authentication phase, the hash of the biometrically encrypted data D_i provided by the election authority should match with $H(D_i)$ value stored in the voter's smart card to ensure that D_i provided by the election authority is correct and unaltered. Suppose a voter's biometrically encrypted data D_i is somehow leaked from the election authority's storage system to an adversary and she attempts a brute-force attack on the Fuzzy Vault system by trying to decode the secret r_{i1}. In that case, if size of the secret key r_{i1} is 128 bit, and the number of chaff and genuine points in the vault are 200 and 24 respectively, then the probability that a combination of points decodes the secret is approximately 4×10^{-10}. The security provided by the biometric encryption system has been discussed in more detail in [22]. Note that besides the secret key r_{i1}, the adversary also needs to know r_{i2} to deduce a token number correctly. Hence, a imposter cannot generate a ballot correctly. Therefore, a correct token can only be generated by a registered voter. This can be verified by the public. Hence, the legitimacy of the voter is ensured.

One-Person-One-Vote without revealing any correspondence between the person and her ballot (encrypted vote). Only one vote per one voter is ensured by the fact that an entry $(VOTER\_ID_i, E(g^{H(r_{i1}\|r_{i2})}))$ on the public bulletin board (Fig. 3) corresponds to only one entry $g^{H(r_{i1}\|r_{i2})}$ on the output of the last mixnet server without revealing the exact correspondence via mixnet servers [64]. The correctness of the shuffling procedure can be verified by checking all the NIZK proofs provided by the mixnet servers [64]. This hides any relation between the voter's voter id $VOTER\_ID_i$ (or voter's name) and $g^{H(r_{i1}\|r_{i2})}$. Hence, it does not reveal any correspondence between the voter and her token number $H(r_{i1}\|r_{i2})$ (say, $token_i$). Fig. 3 illustrates this fact. Now since g^{token_i} and $P_K\{token_i\}$ are included in her ballot on the BB (Fig. 6), one vote per one person is ensured by verifying three facts: g^{token_i} in all confirmed ballots on the BB (Fig. 6) are different; there exists only one entry g^{token_i} on the BB (Fig. 3) generated during the voter registration phase; and the correctness of the NIZK proof $P_K\{token_i\}$. This can also be verified by the public.

Therefore, the proposed scheme satisfies the eligibility verifiability property. Thus, the ballot stuffing attack is prevented.

6.3.2. End-to-end verifiability and integrity of the voting system

In this section, we prove that the proposed system achieves end-to-end verifiability and preserves integrity of the election tally. We show how voter-initiated auditing ensures that the votes are cast as intended and recorded as cast. We also prove that, assuming all well-formedness

proofs are proof of knowledge, if all public verification succeed, votes are tallied as recorded. The number of voters is assumed to be less than the size of the group q.

Voter initiated auditing performs three checks. First, the voter observes that the first part of the receipt is provided before deciding whether to audit or confirm a ballot. Second, if the voter chooses to audit a ballot, she is provided with another receipt reflecting her vote v_i and randomness r_i. Thus, the voter can verify that her vote v_i is correctly captured by the DRE. Third, the voter matches her receipts with those on the BB. Thus, the voter makes sure that her receipts are recorded by the BB without any modification. The public verification of the consistency of the audited ballots guarantees that the DRE has been successful to respond to the challenges made by the voter. Hence, the individual verification (step 7 of the voting phase in case of audit and step 8 of the voting phase) and the public audit consistency verification (step 2(iv) and 2(v) of the Tallying phase) collectively guarantee that the votes are cast as intended and recorded as cast.

In the following theorem, we show that if the ballot well-formedness, signature and tally verification succeed, and hash of each transaction matches with the *prev_hash* field of the next transaction, the proposed system achieves *tallied as recorded* property.

Theorem 1. *In the proposed system, assuming that all proofs of well-formedness are proofs of knowledge, if the public ballot well-formedness, signature, token and tally verifications succeed, and hash of each transaction matches with the prev_hash field of the next transaction, then the reported tally t is the correct tally of all confirmed votes on the bulletin board.*

The proof of the above theorem is rather straightforward and hence omitted here. In the proposed algorithm, any ballot is linked to its previous ballot by using the hash of the previous ballot in the signature of the ballot. This creates an append-only list of valid ballots. Having ballots linked to its previous ballot prevent a malicious bulletin board from adding new ballots. It can be shown how all the public verifications, proof of well-formedness and the first tally verification checks (i.e. first of the two in Eq. (2)) collectively guarantee that the second tally verification (i.e. the second of the two in Eq. (2)) holds if and only if $t = \sum_{i \in \mathbb{C}} v_i$, where $\mathbb{C}$ is the set of all confirmed votes. Hence, if hash of the previous ballot matches with the *prev_hash* field of the current ballot and well-formedness, signature, token number and tally verification succeed, the reported tally t is the correct tally of all confirmed ballots on the BB (Fig. 6).

If the adversary does not get access to both the secret key of the underlying signature algorithm and a valid unused token number *token*, she cannot post a valid ballot on the BB. However, if the adversary gets access to the secret key of the underlying signature algorithm, a valid unused token number *token* and gets read access to the DRE storage variables (for example, partial sum s, partial tally t and *prev_hash*), then she can post adversarial ballot on the BB; however, it will be detected immediately since the DRE will not be able to post the next ballot on the BB. The voter will not be able to match her receipts with those on the BB. Therefore, this attack can be detected immediately. The invalid transactions (adversarial ballots) that are causing this inconsistency can also be determined immediately by observing the *prev_hash* field of the current ballot generated by the DRE. If facilities are available, the bulletin board may remove those invalid transactions from the end of the bulletin board to add the current valid ballot.

Note that, in DRE-ip, if an adversary gets access to the secret key of the underlying signature algorithm, the adversary can post ballots on the bulletin board in such a way that it cannot be detected by the tally verification process (including the public verification and tally verification process) in the tallying phase. A malicious DRE-ip system can also add such valid ballots on the BB. This has been described earlier in Section 4.

6.3.3. Ballot secrecy and the voter's privacy

In this section, we prove that our scheme is secure against all probabilistic polynomial time adversary that try to deduce a particular voter's vote. We show that the public bulletin board does not reveal any information about a voter's vote except what a tally normally does. In an election with n_1 voters, if an attacker colludes with some m_1 number of voters, she will learn the partial tally of the remaining $(n_1 - m_1)$ voters; however, she will not be able to deduce any honest voter's vote. Note that the attacker can compute the partial tally by subtracting the colluding voter's votes from the final tally.

We consider intrusive attacks to the DRE machine in which the adversary gets read access to the DRE storage for a certain period during the voting phase. The adversary is able to observe the vote cast during that access period and also gets read access to the running tally t, partial sum s and the running hash *prev_hash*. The adversary can also read the publicly available information on the bulletin board. We prove that if an adversary can make temporary access to the DRE machine at a certain time T, she will only learn the partial tally of all cast votes from the start of the election to the time T and from the time T to the end of the election.

For the purpose of this proof, we consider an abridged bulletin board where the zero-knowledge proofs of well-formedness are simulated. Assuming that the zero-knowledge proofs are secure, the adversary will only have negligible advantage while dealing with them. In the rest of the proof, we shall not mention the zero-knowledge proofs with each ballot; however, it is provided with each ballot.

Lemma 1. *In our proposed scheme, if an attacker colludes with some m_1 $(m_1 < n_1)$ number of voters and gets read access to the DRE storage after n_1 voters cast their votes, she will only learn the partial tally of $(n_1 - m_1)$ uncompromised voters, not the individual votes of the uncompromised voters.*

Proof. Without any loss of generality, we assume that the indices of the uncompromised (honest) voters are $\{1, 2, \ldots, n_1 - m_1\}$, and that of the colluding voters are $\{n_1 - m_1 + 1, n_1 - m_1 + 2, \ldots, n_1\}$. The DRE stores only four variables: the sum of all random numbers s generated so far for creating the ballots; the running tally t of all cast (confirmed) votes, the running hash *prev_hash* (hash of the previous ballot) and the unique ballot index i. Since s and t are initialized with 0 at the beginning of the election, if the attacker gets read access to the DRE storage after n_1 voters have cast their votes, she will learn the partial tally of n_1 voters and partial sum of randomness used to create their ballots and the running hash *prev_hash* after n_1 ballots. The attacker will know all the colluding voters' vote; however, the randomness used to create their ballots will remain secret to the attacker. This is because, after a vote is cast, the DRE securely deletes the corresponding random number and the vote used to compute the confirmed ballot. Each ballot is of the form (U_i, V_i) along with the unique ballot index i, running hash *prev_hash*, g^{token}, $P_K\{token\}$, NIZK proof of ballot well-formedness and the signature of the ballot, where $U_i = g_1^{r_i}$ and $V_i = g_2^{r_i} g_2^{v_i}$ $\forall i \in \{1, 2, \ldots, n_1\}$. The attacker will know the colluding voters' votes $v_{n_1 - m_1 + 1}, v_{n_1 - m_1 + 2}, \ldots, v_{n_1}$. Hence, she can compute the partial tally of uncompromised voters' vote by subtracting these votes from the overall tally t. However, the randomness r_i will remain secret to the attacker $\forall i \in \{1, 2, \ldots, n_1\}$. The randomness of an uncompromised voter, $r_i = s - \{r_1 + r_2 + \cdots + r_{i-1} + r_{i+1} + \cdots + r_{n_1}\}$. Now all r_i's are uniformly and independently distributed over $\mathbb{Z}_q$. The partial sum s is known to the attacker. Now if at least one of the random values r_j, where $j \in \{1, 2, \ldots, i-1, i+1, \ldots, n_1\}$, is unknown to the attacker, r_i will be a random value unknown to the attacker. Hence, for all uncompromised voters, the value r_i to compute the ballot (U_i, V_i) is random, uniformly distributed over $\mathbb{Z}_q$ and unknown to the attacker even if the attacker knows the partial sum s and the running hash *prev_hash*. Moreover, according to the protocol, the secret key of the encryption i.e. the discrete logarithm between g_1 and g_2 is either not known to any party or securely deleted during the setup stage (see, for instance, [78] for

solution to secure data deletion). Hence, to deduce an uncompromised voter's vote from the ballot (U_i, V_i), the attacker has to solve a discrete logarithm problem (DLP). Therefore, if the discrete logarithm problem (DLP) is hard to solve in the group $\mathbb{G}_q$, an attacker will not be able deduce an uncompromised voter's vote. □

The adversary can control arbitrary number of voters, and in effect she can cast arbitrary number of votes. We call the votes cast or observed by an adversary adversarial votes. The adversary can compute the tally of non-adversarial votes cast before and after the adversarial access period by using the knowledge of adversarial votes, final tally and the partial tally during the adversarial access period.

Theorem 2. *Suppose the election begins at time T_{begin} and finishes at time T_{end}. If an attacker determines arbitrary number of votes and gets temporary read access to the DRE storage during a certain period $[T_0, T_1] \subset [T_{begin}, T_{end}]$, then she will only learn the partial tallies of non-adversarial votes cast between the time T_{begin} to T_0 and between T_1 to T_{end}, but not the individual non-adversarial votes.*

Proof. The proof of this theorem follows from Lemma 1. □

We have proved the theorem for one adversarial access period only. However, it can also be extended to multiple adversarial access period.

Receipt-freeness. We consider a definition of receipt-freeness which requires that a voter cannot produce a receipt to prove that she votes in a particular way (i.e. for a particular candidate). Its purpose is to protect against vote buying. This definition originates from Benaloh [79]. The proposed system provides a receipt to the voter; however, the voter cannot prove that she votes in a particular way since her vote is encrypted. The public key g_1 of the ElGamal encryption algorithm can be generated from the group generator g_2 using a hashing algorithm in such a way that the discrete logarithm (secret key) relationship is not known to any party including the DRE (or this secret key is securely deleted during the setup stage). Furthermore, after encrypting each confirmed vote, the DRE securely deletes corresponding random variable r_i and the vote v_i. These information are not provided to the voter or stored in the DRE machine. Therefore, a voter using her receipts provided by the DRE cannot prove that she voted for a particular candidate. Hence, the proposed scheme is receipt-free.

6.4. Zero-knowledge proofs

In this section, we present the NIZK proof algorithms that is required for well-formedness proof of the ballot. We first recall the NIZK proof algorithms used in DRE-ip [4] and then the efficient NIZK proof algorithm proposed by Lin et al. in [23]. Subsequently, we analyze the efficient NIZK proof protocol proposed in [23] and prove that it does not satisfy the required security properties of a zero-knowledge proof. Thereafter, we propose a NIZK proof algorithm that is more efficient than the NIZK proof presented in DRE-ip [4]. It is shown that our proposed efficient NIZK proof algorithms satisfy all the required security properties of a NIZK proof.

Assume that there are n candidates contesting the election. Let us assume that the vote v_i in the ith ballot is given to the jth candidate, where $j \in \{1, 2, \dots, n\}$. As discussed in Section 6.1.3, we encode the vote for the jth candidate as $v = M^{j-1}$, where $j \in \{1, 2, \dots, n\}$. In this case, the ith ballot will be of the form $((i, g^{token}, prev\_hash, U_i, V_i, P_{WF}\{V_i\}, P_K\{token\}), audited, r_i, v_i, sign)$ in case of audit or $((i, g^{token}, prev\_hash, U_i, V_i, P_{WF}\{V_i\}, P_K\{token\}), confirmed, sign)$ in case of confirmed vote, where $U_i = g_1^{r_i}$ and $V_i = g_2^{r_i} g_2^{M^{j-1}}$. The well-formedness proof $P_{WF}\{V_i\}$ is a 1-out-of-n disjunctive proof and can be stated as:

$$P_{WF}\{V_i : g_1, g_2, U_i\} = P_K\{r_i : \vee_{j=1}^{n}((U_i = g_1^{r_i}) \wedge (V_i / g_2^{M^{j-1}} = g_2^{r_i}))\}$$
$$= P_K\{r_i : (U_i = g_1^{r_i}) \wedge (\vee_{j=1}^{n}(V_i / g_2^{M^{j-1}} = g_2^{r_i}))\}.$$

6.4.1. Revisiting the 1-out-of-n NIZK proof used in DRE-ip

Algorithm 1 (resp. Algorithm 2) represents the prover algorithm (resp. verifier algorithm) for generation (resp. verification) of the 1-out-of-n NIZK proof used in DRE-ip. This 1-out-of-n NIZK proof is extended from the 1-out-of-2 NIZK proof given in [4]. Algorithm 1 (resp. Algorithm 2) is written to prove (resp. verify) a proposition of the form $\vee_{l=1}^{n}((\Gamma' = \{\gamma'\}^{\lambda}) \wedge (\Gamma_l'' = \{\gamma_l''\}^{\lambda}))$.

Algorithm 1: A prover with identifier ID generates a proof of knowledge of a secret λ such that $\vee_{l=1}^{n}((\Gamma' = \{\gamma'\}^{\lambda}) \wedge (\Gamma_l'' = \{\gamma_l''\}^{\lambda}))$ for known $ID, n, \Gamma', \gamma', (\Gamma_l'', \gamma_l'')_{l=1}^{n}$, where the vote is given to the jth candidate, $j \in \{1, 2, \dots, n\}$.

Input : $ID, n, \Gamma', \gamma', (\Gamma_l'', \gamma_l'')_{l=1}^{n}, \lambda, j$ such that $\Gamma' = \{\gamma'\}^{\lambda}$ and $\Gamma_j'' = \{\gamma_j''\}^{\lambda}$
Output: $\Pi = P_K\{\lambda : \vee_{l=1}^{n}((\Gamma' = \{\gamma'\}^{\lambda}) \wedge (\Gamma_l'' = \{\gamma_l''\}^{\lambda}))\}$
begin
  choose random
  $w, r_1, c_1, r_2, c_2, \dots, r_{j-1}, c_{j-1}, r_{j+1}, c_{j+1}, \dots, r_n, c_n \in \mathbb{Z}_q$
  calculate $t_{11} = \{\gamma'\}^{r_1}\{\Gamma'\}^{c_1}, t_{12} = \{\gamma_1''\}^{r_1}\{\Gamma_1''\}^{c_1}, t_{21} = \{\gamma'\}^{r_2}\{\Gamma'\}^{c_2}, t_{22} = \{\gamma_2''\}^{r_2}\{\Gamma_2''\}^{c_2}, \dots, t_{j-11} = \{\gamma'\}^{r_{j-1}}\{\Gamma'\}^{c_{j-1}}, t_{j-12} = \{\gamma_{j-1}''\}^{r_{j-1}}\{\Gamma_{j-1}''\}^{c_{j-1}}, t_{j1} = \{\gamma'\}^{w}, t_{j2} = \{\gamma_j''\}^{w}, t_{j+11} = \{\gamma'\}^{r_{j+1}}\{\Gamma'\}^{c_{j+1}}, t_{j+12} = \{\gamma_{j+1}''\}^{r_{j+1}}\{\Gamma_{j+1}''\}^{c_{j+1}}, \dots, t_{n1} = \{\gamma'\}^{r_n}\{\Gamma'\}^{c_n}, t_{n2} = \{\gamma_n''\}^{r_n}\{\Gamma_n''\}^{c_n}$
  calculate
  $c = H(ID, (\gamma', \Gamma', \gamma_l'', \Gamma_l'')_{l=1}^{n}, (t_{l1}, t_{l2})_{l=1}^{n})$,
  calculate
  $c_j = c - (c_1 + c_2 + \cdots + c_{j-1} + c_{j+1} + \cdots + c_n)$
  calculate $r_j = w - c_j\lambda$
  return $\Pi = (c_1, c_2, \dots, c_{j-1}, c_j, c_{j+1}, \dots, c_n, r_1, r_2, \dots, r_{j-1}, r_j, r_{j+1}, \dots, r_n)$
end

Algorithm 2: Verification of proof Π generated by Algorithm 1 given $ID, n, \Gamma', \gamma', (\Gamma_l'', \gamma_l'')_{l=1}^{n}$. However, the verifier does not know to which candidate (i.e. j) the vote is given.

Input : $ID, n, \Gamma', \gamma', (\Gamma_l'', \gamma_l'')_{l=1}^{n}, \Pi = (c_1, c_2, \dots, c_n, r_1, r_2, \dots, r_n)$
Output: success or failure
begin
  calculate
  $t_{11} = \{\gamma'\}^{r_1}\{\Gamma_1'\}^{c_1}, t_{12} = \{\gamma_1''\}^{r_1}\{\Gamma_1''\}^{c_1}, t_{21} = \{\gamma'\}^{r_2}\{\Gamma_2'\}^{c_2}, t_{22} = \{\gamma_2''\}^{r_2}\{\Gamma_2''\}^{c_2}, \dots,$
  $t_{n1} = \{\gamma'\}^{r_n}\{\Gamma_n'\}^{c_n}, t_{n2} = \{\gamma_n''\}^{r_n}\{\Gamma_n''\}^{c_n}$
  calculate
  $c' = H(ID, (\gamma', \Gamma', \gamma_l'', \Gamma_l'')_{l=1}^{n}, (t_{l1}, t_{l2})_{l=1}^{n})$
  if $c' = (c_1 + c_2 + \cdots + c_n)$ **then**
    return success
  else
    return failure
  end
end

6.4.2. Revisiting the efficient 1-out-of-n NIZK proof proposed by Lin et al.

In [23], Lin et al. proposed a 1-out-of-n NIZK proof and claimed that it is more efficient than the original NIZK proof. We describe the 1-out-of-n NIZK proof proposed by Lin et al. [23] almost verbatim. We then analyze this NIZK proof and show that it does not satisfy the properties of zero-knowledge proof. The 1-out-of-n NIZK proof algorithm is described considering our case.

1-out-of-n NIZK proof by Lin et al. The prover and the verifier are divided into two parts respectively depending on the parity of j (even or odd), where the vote is given to the jth candidate.

The prover chooses a random number w. If j is even, the prover chooses random numbers $r_\beta, d_\beta, \forall\beta \in \{2,4,\dots,n\}$; otherwise, if j is odd, the prover chooses random numbers $r_\eta, d_\eta, \forall\eta \in \{1,3,\dots,n-1\}$. Thereafter, the prover calculates the vote $(U_i, V_i) = (g_1^{r_i}, g_2^{r_i} g_2^{M^{j-1}})$ and $(t_{j1}, t_{j2}) = (g_1^w, g_2^w)$. The prover then calculates $(t_{l1}, t_{l2}) = (g_1^{r_l} U_i^{d_l}, g_2^{r_l}\{V_i/g_2^{M^{j-1}}\}^{d_l})$, where $l = 2,4,\dots,n$ if j is even or $l = 1,3,\dots,n-1$ if j is odd. For non-interactiveness, $c_{all} = H(r_i\|U_i\|V_i)$. Then the prover computes $c_{even} = H(r_i\|U_i\|V_i\|\{t_{l1},t_{l2}\}_{l=2(l\in even)}^{n})$ if j is even or $c_{odd} = H(r_i\|U_i\|V_i\|\{t_{l1},t_{l2}\}_{l=1(l\in odd)}^{n-1})$ if j is odd.

Then the prover computes the following parameters based on the parity of j.

If j is even, the prover calculates $(t_{\beta1}, t_{\beta2}) = (g_1^{r_\beta} U_i^{d_\beta}, g_2^{r_\beta}\{V_i/g_2^{M^{\beta-1}}\}^{d_\beta}), c_{odd} = c_{all} - c_{even}, d_j = c_{even} - \sum_\beta d_\beta, r_j = w - r_i d_j$.

If j is odd, the prover calculates $(t_{\eta1}, t_{\eta2}) = (g_1^{r_\eta} U_i^{d_\eta}, g_2^{r_\eta}\{V_i/g_2^{M^{\eta-1}}\}^{d_\eta}), c_{even} = c_{all} - c_{odd}, d_j = c_{odd} - \sum_\eta d_\eta, r_j = w - r_i d_j$.

Finally, the verifier sends $(\{t_{l1},t_{l2},d_l,r_l\}_{l=2(l\in even)}^{n}, U_i, V_i, c_{odd})$ to the verifier if j is even or $(\{t_{l1},t_{l2},d_l,r_l\}_{l=1(l\in odd)}^{n-1}, U_i, V_i, c_{even})$ to the verifier if j is odd.

The verifier verifies the correctness of c_{even} or $c_{odd} = \sum_l d_l$, $c_{all} = c_{even} + c_{odd}$ and $(t_{l1}, t_{l2}) = (g_1^{r_l} U_i^{d_l}, g_2^{r_l}\{V_i/g_2^{M^{l-1}}\}^{d_l})$, where $l = \{2,4,\dots,n\}$ $or\{1,3,\dots,n-1\}$. If the verifier verifies these conditions successfully, it returns success; otherwise, it returns failure.

Analysis of this 1-out-of-n NIZK proof. According to the protocol, the verifier has to compute $H(r_i\|U_i\|V_i)$ or $H(r_i\|U_i\|V_i\|\{t_{l1},t_{l2}\}_{l=2(l\in even)}^{n})$ if j is even or $H(r_i\|U_i\|V_i\|\{t_{l1},t_{l2}\}_{l=1(l\in odd)}^{n-1})$ if j is odd to verify the conditions: c_{even} or $c_{odd} = \sum_l d_l$, $c_{all} = c_{even} + c_{odd}$. However, the verifier cannot compute these hash functions since she does not know the secret value r_i used in the argument of the hash function. Therefore, this 1-out-of-n NIZK proof does not satisfy the completeness property of the zero-knowledge proof.

Moreover, according to the protocol, the verifier has to verify the conditions $(t_{l1}, t_{l2}) = (g_1^{r_l} U_i^{d_l}, g_2^{r_l}\{V_i/g_2^{M^{l-1}}\}^{d_l})$, where $l = \{2,4,\dots,n\}$ $or\{1,3,\dots,n-1\}$. To verify these conditions, the verifier has to know whether j is even or odd. It means that the verifier has to know whether the voter has given her vote to an even numbered candidate or an odd numbered candidate. In other words, if the verifier verifies it successfully when $l = \{2,4,\dots,n\}$, the verifier will know that the voter has given her vote to an even numbered candidate (since j is even). Similarly, if the verifier verifies it successfully when $l = \{1,3,\dots,n-1\}$, the verifier will know that the voter has given her vote to an odd numbered candidate (since j is odd). Therefore, this 1-out-of-n NIZK proof does not satisfy the witness indistinguishability property of the zero-knowledge proof.

6.4.3. Our proposed efficient 1-out-of-n NIZK proof

We propose an efficient 1-out-of-n NIZK proof. Our proposed 1-out-of-n NIZK proof satisfy all the required security properties of the zero-knowledge proof. The security proofs of the proposed 1-out-of-n NIZK proof are given in Appendix B. We have modified the zero-knowledge proof involving the conjunction and disjunction of predicates to improve its performance. The 1-out-of-n (Algorithms 3 and 4) proofs presented here are efficient than the proofs presented in DRE-ip [4]. Algorithm 3 (resp. Algorithm 4) is written to prove (resp. verify) a proposition of the form $\vee_{l=1}^{n}((\Gamma' = \{\gamma'\}^\lambda) \wedge (\Gamma_l'' = \{\gamma_l''\}^\lambda))$ which is equivalent to $(\Gamma' = \{\gamma'\}^\lambda) \wedge (\vee_{l=1}^{n}(\Gamma_l'' = \{\gamma_l''\}^\lambda))$. As we assumed previously, the vote is given to the jth candidate. We assume that the simultaneous multiple exponentiation (SME) [80] technique is used to optimize the computation of a term of the form $g^x h^y$. The computation cost of the term $g^x h^y$ is around 1.2 exponentiation using SME technique. The prover algorithm (Algorithm 3) requires $(1.2(n-1)+2)$ exponentiations; however, the prover algorithm (Algorithm 1) presented in DRE-ip requires $(2.4(n-1)+2)$ exponentiations. To verify such zero-knowledge proof, the verifier algorithm (Algorithm 4) requires $1.2(n+1)$ exponentiations; however, the verification algorithm (Algorithm 2) presented in DRE-ip requires $2.4n$ exponentiations. In Table 4, we theoretically analyze the cost of execution of the prover and the verifier of the original 1-out-of-n NIZK proof and our proposed efficient 1-out-of-n NIZK proof. Since exponential operations are the most time consuming operations, we only include the number of exponentiations in theoretical analysis (Table 4).

Algorithm 3: A prover with identifier ID generates a proof of knowledge of a secret λ such that $(\Gamma' = \{\gamma'\}^\lambda) \wedge (\vee_{l=1}^{n}(\Gamma_l'' = \{\gamma_l''\}^\lambda))$ for known $ID, n, \Gamma', \gamma', (\Gamma_l'', \gamma_l'')_{l=1}^{n}$, where the vote is given to the jth candidate, $j \in \{1,2,\dots,n\}$.

Input : $ID, n, \Gamma', \gamma', (\Gamma_l'', \gamma_l'')_{l=1}^{n}, \lambda, j$ such that $\Gamma' = \{\gamma'\}^\lambda$ and $\Gamma_j'' = \{\gamma_j''\}^\lambda$
Output: $\Pi = P_K\{\lambda : (\Gamma' = \{\gamma'\}^\lambda) \wedge (\vee_{l=1}^{n}(\Gamma_l'' = \{\gamma_l''\}^\lambda))\}$
begin
  choose random
  $w, r_1, c_1, r_2, c_2, \dots, r_{j-1}, c_{j-1}, r_{j+1}, c_{j+1}, \dots, r_n, c_n \in \mathbb{Z}_q$
  calculate $w_1 = w + (r_1 + r_2 + \cdots + r_{j-1} + r_{j+1} + \cdots + r_n) + (c_1 + c_2 + \cdots + c_{j-1} + c_{j+1} + \cdots + c_n)\lambda$
  calculate $t_1 = \{\gamma'\}^{w_1}, t_{12} = \{\gamma_1''\}^{r_1}\{\Gamma_1''\}^{c_1}, t_{22} = \{\gamma_2''\}^{r_2}\{\Gamma_2''\}^{c_2}, \dots, t_{j-12} = \{\gamma_{j-1}''\}^{r_{j-1}}\{\Gamma_{j-1}''\}^{c_{j-1}}, t_{j2} = \{\gamma_j''\}^{w}, t_{j+12} = \{\gamma_{j+1}''\}^{r_{j+1}}\{\Gamma_{j+1}''\}^{c_{j+1}}, \dots, t_{n2} = \{\gamma_n''\}^{r_n}\{\Gamma_n''\}^{c_n}$
  calculate
  $c = H(ID, \gamma', \Gamma', (\gamma_l'', \Gamma_l'')_{l=1}^{n}, t_1, (t_{l2})_{l=1}^{n})$,
  calculate
  $c_j = c - (c_1 + c_2 + \cdots + c_{j-1} + c_{j+1} + \cdots + c_n)$
  calculate $r_j = w - c_j\lambda$
  return $\Pi = (c_1, c_2, \dots, c_{j-1}, c_j, c_{j+1}, \dots, c_n, r_1, r_2, \dots, r_{j-1}, r_j, r_{j+1}, \dots, r_n)$
end

Algorithm 4: Verification of proof Π generated by Algorithm 3 given $ID, n, \Gamma', \gamma', (\Gamma_l'', \gamma_l'')_{l=1}^{n}$. However, the verifier does not know to which candidate (i.e. j) the vote is given.

Input : $ID, n, \Gamma', \gamma', (\Gamma_l'', \gamma_l'')_{l=1}^{n}, \Pi = (c_1, c_2, \dots, c_n, r_1, r_2, \dots, r_n)$
Output: success or failure
begin
  calculate
  $t_1 = \{\gamma'\}^{r_1+r_2+\cdots+r_n}\{\Gamma_1'\}^{c_1+c_2+\cdots+c_n}, t_{12} = \{\gamma_1''\}^{r_1}\{\Gamma_1''\}^{c_1}, t_{22} = \{\gamma_2''\}^{r_2}\{\Gamma_2''\}^{c_2}, \dots,$
  $t_{n2} = \{\gamma_n''\}^{r_n}\{\Gamma_n''\}^{c_n}$
  calculate
  $c' = H(ID, \gamma', \Gamma', (\gamma_l'', \Gamma_l'')_{l=1}^{n}, t_1, (t_{l2})_{l=1}^{n})$
  if $c' = (c_1 + c_2 + \cdots + c_n)$ **then**
    return success
  else
    return failure
  end
end

Table 4
Computation complexity of the 1-out-of-n NIZK and the proposed 1-out-of-n NIZK proof. e represents the exponentiation operation.

Scheme	Prover	Verifier
1-out-of-n NIZK proof	$(2.4(n-1)+2)e$	$2.4ne$
Proposed 1-out-of-n NIZK proof	$(1.2(n-1)+2)e$	$1.2(n+1)e$

Table 5
Security assumptions for some DRE-based verifiable e-voting systems. Columns are represented as — A: Reliable Tallying authorities, B: Sufficient Voter-initiated auditing, C: Protection against malicious bulletin board, D: Secure setup, E: Secure random number generator, F: Secure Deletion, G: Secure Ballot Storage, H: Trust-worthy tallying authorities, I: Secure computation (with proof of correctness) of the final tally without revealing the results from each DRE machine when multiple DRE machines are used, J: secure voter registration and authentication. •: assumption is required, ∘: assumption is not required.

System	A	B	C	D	E	F	G	H	I	J
Votegrity	•	•	•	•	•	•	∘	•	•	•
MarkPledge	•	•	•	•	•	•	∘	•	•	•
VoteBox	•	•	•	•	•	•	∘	•	•	•
STAR-Vote	•	•	•	•	•	•	∘	•	•	•
DRE-i	∘	•	•	•	•	•	•	∘	•	•
vVote	•	•	•	•	•	•	∘	•	•	•
DRE-ip	∘	•	•	•	•	•	∘	∘	•	•
Proposed system using cloud server/IPFS	∘	•	∘	•	•	•	∘	∘	∘	∘
Proposed system using blockchain (method 1)	∘	•	∘	•	•	•	∘	∘	∘	∘
Proposed system using both cloud/IPFS and blockchain (method 2)	∘	•	∘	•	•	•	∘	∘	∘	∘

These algorithms can be extended to prove any proposition of the form $\wedge_{i=1}^{k}\varphi_i \wedge (\vee_{l=1}^{n}\psi_l)$ for a set of assertions $\{\varphi_1, \varphi_2, \ldots, \varphi_k, \psi_1, \psi_2, \ldots, \psi_n\}$, where the numbers k and n are known to both the prover and the verifier. To generate such zero-knowledge proof, the proposed prover algorithm (extended version of Algorithm 3) requires $(1.2(n-1)+k+1)$ exponentiations; however, the prover algorithm (extended version of Algorithm 1) presented in DRE-ip requires $(1.2(k+1)(n-1)+k+1)$ exponentiations. To verify such zero-knowledge proof, the proposed verifier algorithm (extended version of Algorithm 4) requires $1.2(n+k)$ exponentiations; however, the verification algorithm (extended version of Algorithm 2) presented in DRE-ip requires $1.2n(k+1)$ exponentiations. Therefore, the proposed NIZK proofs are almost $(k+1)$ times more efficient than the NIZK proofs presented in [4].

6.5. Comparison

In this section, we discuss how our proposed system compares with other DRE-based verifiable e-voting systems. In particular, we compare with Chaum's Votegrity [1], Neff's MarkPledge [24], VoteBox [72], Star-Vote [26], DRE-i [3], vVote [32], and DRE-ip [4]. All of these systems consider voter registration and voter authentication outside of their scope and assume that they are performed securely and correctly. Our scheme comprises a secure and verifiable voter registration and authentication mechanism. All of the above mentioned systems rely on a secure bulletin board. In our proposed system, if the BB is insecure, it will be detected by the public or individual voters during the voting phase or tallying phase. The ballots can be stored using in any of these methods: cloud server, IPFS, blockchain (method 1), both cloud server/IPFS and blockchain (method 2). A comparison of these systems in terms of their underlying security assumptions is given in Table 5 (also see DRE-ip [4] for security assumptions of these systems).

We now discuss the computational complexity of different DRE-based e-voting systems and compare it with our proposed system. We do not consider Votegrity, MarkPledge, and vVote since they use mixnets and the computational complexity depends on the implementation of those mixnets. Let us compute all calculations based on a two-candidate election, encryption based on ElGamal cryptosystem and one tallying authority (TA) if present. If the number of TAs increases, the complexity of tally calculations and verification of all the systems requiring tallying authorities also increases. We assume that the TA, if present, provides proof of correctness as required by the end-to-end verifiability. We also assume that the simultaneous multiple exponentiation (SME) [80] technique is used for optimization. Using this technique, the computation cost of the term $g^x h^y$ is equivalent to around 1.2 exponentiations. The voter authentication process and the voting phase can be performed in pipeline. Table 6 summarizes the computational complexity of different systems (also see DRE-ip [4] for performance comparison of these systems).

The ElGamal encryption for a ballot takes around 2 exponentiations. The zero-knowledge proof $P_{WF}\{V_i\}$ takes 3.2 exponentiations to generate and 3.6 exponentiations to verify using the proposed efficient NIZK proof Algorithm 3 (for generation) and Algorithm 4 (for verification) described in Section 6.4. Therefore, a ballot creation requires 5.2 exponentiations for both audited and confirmed ballot. Since our system comprises a secure and verifiable voter authentication mechanism, this introduces some additional computations in our system. The computation of g^{token} requires one exponentiation. The zero-knowledge proof $P_K\{token\}$ takes 1 exponentiation to generate and 1.2 exponentiations to verify using the NIZK proof Algorithm 5 (for generation) and Algorithm 6 (for verification) described in Appendix A. Therefore, a ballot creation requires about 7.2 exponentiations for both audited and confirmed ballot. The ballot well-formedness and consistency verification take about 6.8 exponentiations and 4.8 exponentiations for audited ballot and confirmed ballot respectively. The tally calculation and tally verification require about 8 exponentiations and $(2|\mathbb{C}|+1)m+4.8e$ computations respectively, where 'm' and 'e' denote the multiplication and exponentiation respectively.

Table 6
Computation complexity of some DRE-based verifiable e-voting systems assuming two-candidate election. Columns are represented as — A: Ballot calculation, B: Ballot well-formedness and consistency verification, C: Tally calculation, D: Tally verification. $\mathbb{B}, \mathbb{A}, \mathbb{C}$ represent all, audited and confirmed ballots respectively. e: exponentiation and m: multiplication.

System	A	B	C	D
VoteBox	$6.4\|\mathbb{B}\|e$	$(6.8\|\mathbb{A}\|+4.8\|\mathbb{C}\|)e$	$\|\mathbb{C}\|m+3e$	$\|\mathbb{C}\|m+2.4e$
STAR-Vote	$6.4\|\mathbb{B}\|e$	$(6.8\|\mathbb{A}\|+4.8\|\mathbb{C}\|)e$	$\|\mathbb{C}\|m+3e$	$\|\mathbb{C}\|m+2.4e$
DRE-i	$10.8\|\mathbb{B}\|e$	$(9.6\|\mathbb{A}\|+4.8\|\mathbb{C}\|)e$		$\|\mathbb{B}\|m+1e$
DRE-ip	$6.4\|\mathbb{B}\|e$	$(6.8\|\mathbb{A}\|+4.8\|\mathbb{C}\|)e$		$2\|\mathbb{C}\|m+2e$
Proposed system	$5.2\|\mathbb{B}\|e$	$(5.6\|\mathbb{A}\|+3.6\|\mathbb{C}\|)e$	$8e$	$(2\|\mathbb{C}\|+1)m+4.8e$

However, in case of $n(n \geq 2)$ candidates, the zero-knowledge proof $P_{WF}\{V_i\}$ takes $(1.2(n-1)+2)$ $(=(1.2n+.8))$ exponentiations to generate and $1.2(n+1)$ exponentiations to verify using the proposed efficient NIZK proof Algorithm 3 (for generation) and Algorithm 4 (for verification) given in Section 6.4. Therefore, in this case, ballot calculation requires $(1.2n+4.8)$ exponentiations for both an audited and confirmed ballot including the computations introduced due to voter authentication. Table 7 summarizes the computational complexity of DRE-ip (without voter authentication) and our proposed system (with voter authentication) in case of n candidates.

When multiple DRE machines are deployed for casting votes in an electoral constituency (from where a candidate is to be elected), during the tallying phase, our proposed voting scheme additionally performs two secure multi-party computations. Let us analyze the performance of our proposed secure multi-party computation protocol when the number of DRE machines deployed in an electoral constituency is d. There are only two rounds in our proposed secure multi-party computation. As discussed in [73], minimum two rounds are required to compute a function securely in any secure multi-party computation protocol. In the first round, each DRE machine sends $(d-1)$ random numbers (using symmetric key encryption over an authenticated channel) to the

Table 7
Computation complexity of DRE-ip (without voter authentication) and our proposed e-voting systems (with voter authentication) while supporting voting for 1 out of n candidates ($n \geq 3$). Columns are represented as — A: Ballot calculation, B: Well-formedness and consistency verification, C: Tally calculation, D: Tally verification. $\mathbb{B}, \mathbb{A}, \mathbb{C}$ represent all, audited and confirmed ballots respectively. e: exponentiation and m: multiplication.

System (multiple candidates)	A	B	C	D
DRE-ip	$(2.4n + 1.6)\|\mathbb{B}\|e$	$((2.4n + 2)\|\mathbb{A}\| + 2.4n\|\mathbb{C}\|)e$		$2\|\mathbb{C}\|m + 2e$
Proposed system	$(1.2n + 4.8)\|\mathbb{B}\|e$	$((1.2n + 4.4)\|\mathbb{A}\| + 1.2(n + 2)\|\mathbb{C}\|)e$	$8e$	$(2\|\mathbb{C}\| + 1)m + 4.8e$

Table 8
Computation and communication cost (space) of the proposed secure multi-party computation when the number of DRE machine is d. a is the size of each element of the group $\mathbb{Z}_q$.

	First round	Second round
Computation cost	Each DRE performs $2d$ addition operations on numbers in $\mathbb{Z}_q$	The $(d+1)$th party performs d addition operations on numbers in $\mathbb{Z}_q$
Communication cost (space)	$((d-1) \times a)$ for each DRE	a for each DRE

remaining $(d-1)$ DRE machines and performs $2d$ addition operations on numbers in $\mathbb{Z}_q$ (to compute $t_{id} = (t_i - t_{i1} - t_{i2} - \cdots - t_{id-1})$ and $T_i = t_{1i} + t_{2i} + t_{3i} + \cdots + t_{di}, i \in \{1, 2, \ldots, d\}$). In the second round, the ith DRE sends T_i to the $(d+1)$th party, and then $(d+1)$th party performs d addition operations on numbers in $\mathbb{Z}_q$ (to compute $T = T_1 + T_2 + T_3 + \cdots + T_d$). Therefore, the total number of rounds of the proposed secure multi-party computation is 2. The total computational time for each DRE machine is $2d$ addition operations on numbers in $\mathbb{Z}_q$, where d is the number of DRE machines deployed in the electoral constituency (from where a candidate is to be elected such as a district, not the whole country). The total space required (communication complexity) is $(d \times a)$ bits (for sending d random numbers in $\mathbb{Z}_q$) for each DRE machine, where a is the size of each element of $\mathbb{Z}_q$ in bit. Table 8 highlights the computation and communication cost (space) for the proposed secure multi-party computation when the number of DRE machine is d.

6.6. Performance analysis

6.6.1. Experiment on ethereum (only for method 1)

We deployed our implementation on Ethereum's private network that mimics the production network. We tested our implementation for different number of candidates contesting the election. Two different experiments were performed based on how we use the blockchain.

First, we use blockchain only as a ballot box. The blockchain stores the ballots after verifying its signature without verifying any NIZK proofs. Note that verification of the NIZK proofs by the blockchain is optional (i.e. not necessary) since it can be verified by the public using a separate module. We have carried out experiments with 2, 3, 4, 5, 6 and 7 candidates and plotted the results to show how the costs for casting a ballot vary with different number of candidates. Fig. 7(a) (resp. Fig. 7(b)) highlights the average gas consumption cost (resp. cost in US dollar) for casting a ballot based on different number of candidates. In this experiment, we have calculated the cost in US dollar (denoted by '$' in Fig. 7(b)) and rounded it to three decimal places.

Secondly, we have used blockchain as a ballot box as well as to verify the 1-out-of-n NIZK proof corresponding to the ballot well-formedness proof. The blockchain stores the ballots after verifying its signature as well as the 1-out-of-n NIZK proof. We have performed experiments with 2, 3, 4, 5, 6, 7, 8 and 9 candidates using our proposed efficient 1-out-of-n NIZK ballot well-formedness proof (Algorithms 3 and 4) and plotted the results to show how the costs for casting a ballot vary with different number of candidates. We have also tested with 2, 3, 4, 5 candidates using the original 1-out-of-n NIZK ballot well-formedness proof (Algorithms 1 and 2) and plotted the results to compare the performance. Fig. 8(a) depicts the average gas consumption cost for casting a ballot based on the number of candidates competing in the election, while using the original 1-out-of-n NIZK ballot well-formedness proof and our proposed efficient 1-out-of-n NIZK ballot well-formedness proof. Casting a ballot involves verifying the 1-out-of-n NIZK ballot well-formedness proof, verifying the signature and storing the ballot into the blockchain. From this figure, we see that the proposed 1-out-of-n NIZK proof is about twice more efficient than the original 1-out-of-n NIZK proof used in [4]. The maximum gas capacity that an Ethereum block can consume is about 8 million as in July, 2020. From the figure, we see that each transaction for casting a ballot reaches the computation and storage limit for about 9 candidates while using our proposed efficient 1-out-of-n NIZK proof; whereas, it reaches the maximum gas limit for about 5 candidates while using the original 1-out-of-n NIZK proof. Fig. 8(b) shows the costs for casting a ballot based on the number of candidates competing in the election and while using the original 1-out-of-n NIZK ballot well-formedness proof and our proposed efficient 1-out-of-n NIZK ballot well-formedness proof. In this experiment, we have calculated the costs in US dollar (denoted by '$' in Fig. 8(b)) and rounded it to two decimal places.

6.6.2. Timing analysis (in case of using cloud server, using IPFS, method 1 and method 2)

We implemented the proposed 1-out-of-n NIZK ballot well-formedness proof which is the most time consuming part for generation and verification of a ballot. Fig. 9 depicts the timing analysis measurements for generation of 1-out-of-n NIZK ballot well-formedness proof using our proposed NIZK proof (Algorithm 3) as well as using the original NIZK proof (Algorithm 1), where n is the number of candidates contesting the election. Fig. 10 shows the timing measurement analysis for verification of 1-out-of-n NIZK ballot well-formedness proof using the proposed NIZK proof (Algorithm 4) as well as using the original NIZK proof (Algorithm 2). All tests were performed on a HP Laptop running Windows 8.1 equipped with 2 cores, 1.8 GHz Intel Core i3 and 4 GB RAM. All time measurements are rounded up to the next whole millisecond. We implemented the protocol over an elliptic curve. The time to create and verify a ballot depends on the number of candidates competing in the election. However, it is independent of the number of voters.

To see how the time for generation and verification of the 1-out-of-n NIZK proof using the proposed NIZK proof vary with different number of candidates, we have carried out experiments with 2, 4, 6, ..., 24 candidates. Fig. 9 (resp. Fig. 10) highlights that the computation time to create (resp. verify) the 1-out-of-n NIZK ballot well-formedness proof using the proposed algorithm is almost reduced to half of the time required to create (resp. verify) that using the original NIZK proof algorithm given in [4].

We have also conducted experiments to see how the computation time for verification of the tally vary with different number of voters and different number of candidates contesting the election. We have performed experiments with 50, 100, 200, 400, 600, 800 and 1000 voters and with 2, 3, 4, 5 and 6 candidates and plotted the results. Fig. 11 represents the computation time (in millisecond) for verification of the tally with respect to the number of candidates and the number of voters. The verification time during the tallying phase involves the time to verify the 1-out-of-n NIZK proof, hashchain and the tally verification equation (2).

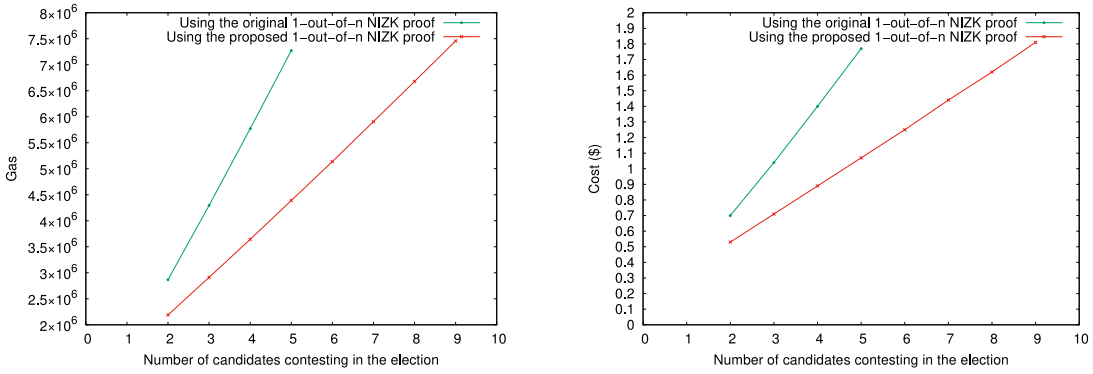

Fig. 8. (a) Gas cost for casting a ballot based on the number of candidates contesting the election while using the original 1-out-of-n NIZK proof and our proposed 1-out-of-n NIZK proof. (b) Costs for casting a ballot based on the number of candidates contesting the election while using the original 1-out-of-n NIZK proof and our proposed 1-out-of-n NIZK proof. The costs are approximated in USD ($) using the conversion rate of 1 Ether = $243 and the gas price of 0.000000001 ether that are real world costs in July, 2020. Here, for both (a) and (b), casting a ballot involves storing the ballot after verification of the 1-out-of-n NIZK proof by the blockchain.

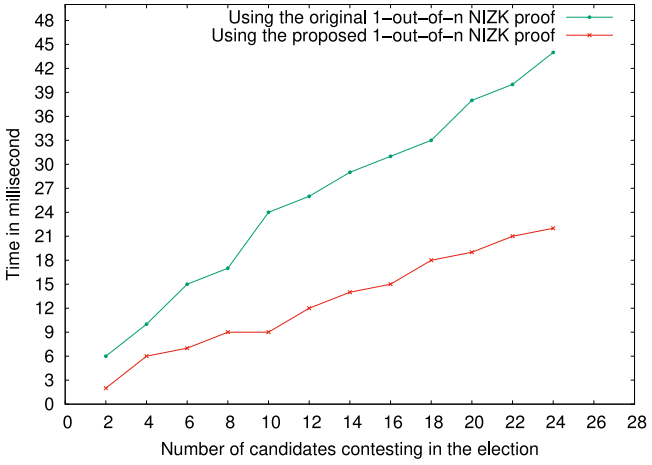

Fig. 9. Computation time to create the 1-out-of-n NIZK proof using the proposed algorithm and the original NIZK algorithm.

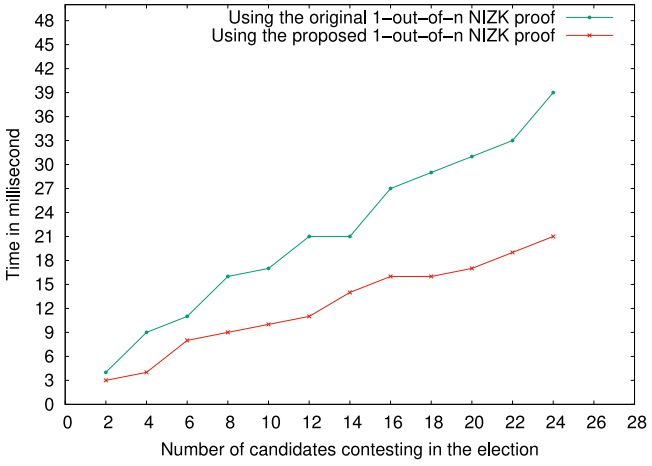

Fig. 10. Computation time for verification of the 1-out-of-n NIZK proof using the proposed algorithm and the original NIZK algorithm.

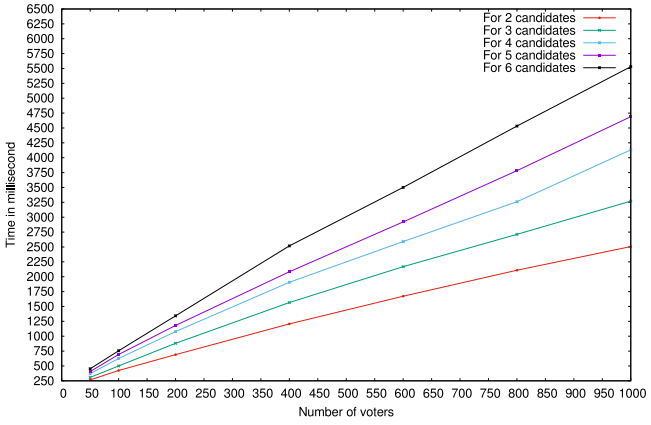

Fig. 11. Computation time for verification of the tally based on the number of voters participating in the election and the number of candidates contesting the election.

7. Conclusion

In this article, we have proposed a secure and verifiable voter registration and authentication mechanism. Thereafter, we have proposed an end-to-end verifiable DRE-based voting system that preserves voter's privacy and integrity of ballots without any tallying authority or secure hardware storage even if the adversary gets temporary access to the DRE machine. The system prevents the well-known ballot stuffing attack and a weakness of the DRE-ip system. Depending on how the election is arranged, we have proposed two methods to store the ballots using blockchain and cloud/IPFS server. We have presented security proofs to prove the security properties of the protocol. We have proposed an efficient 1-out-of-n NIZK proof. Both the theoretic analysis and the experimental results show that the scheme is feasible to be used in practice. One of the challenging tasks in our proposed scheme is to reduce the False Rejection Rate of the Fuzzy Vault algorithm for fingerprint. As previously mentioned, another challenging task is the management of the key used by the Mixnet server. In future work, we plan to design biometric encryption algorithm with improved False Rejection Rate as well as to design DRE-based voting solution without tallying authorities for more complex voting systems such as single transferable vote (STV) [31] and Condorcet.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This research was partially supported by the Royal Society under Grant ICA/R1/180226. The authors would like to thank Dr. Feng Hao, Professor of Security Engineering, University of Warwick, UK, for his advice and comments during the course of this research.

Appendix A

Algorithm 5: A prover with identifier ID generates a NIZK proof of knowledge of a secret x such that $(\Gamma' = g^x)$ for known ID, Γ', g and the encrypted vote (U_i, V_i).

Input : $ID, \Gamma', g, x, U_i, V_i$ such that $(\Gamma' = g^x)$
Output: $\eta = P_K\{x : (\Gamma' = g^x)\}$
begin
  choose random $w \in \mathbb{Z}_q$
  calculate
  $t_1 = g^w$.
  calculate
  $c = H(ID, U_i, V_i, g, \Gamma', t_1)$
  calculate $r = w - cx$
  return $\eta = (c, r)$
end

Algorithm 6: Verification of proof η generated by Algorithm 5 given ID, Γ', g and the encrypted vote (U_i, V_i).

Input : $ID, \Gamma', g, U_i, V_i, \eta = (c, r)$
Output: success or failure
begin
  calculate
  $t_1 = g^r \Gamma'^c$
  calculate
  $c_1 = H(ID, U_i, V_i, g, \Gamma', t_1)$
  if $c_1 = c$ **then**
    return success
  else
    return failure
  end
end

Algorithm 7: A prover with identifier ID generates a NIZK proof of knowledge of a secret x such that $((\Gamma_1 = g_1^x) \wedge (\Gamma_2 = g_2^x))$ for known $ID, \Gamma_1, g_1, \Gamma_2, g_2$.

Input : $ID, \Gamma_1, g_1, \Gamma_2, g_2, x$ such that $((\Gamma_1 = g_1^x) \wedge (\Gamma_2 = g_2^x))$
Output: $\eta = P_K\{x : ((\Gamma_1 = g_1^x) \wedge (\Gamma_2 = g_2^x))\}$
begin
  choose random $w \in \mathbb{Z}_q$
  calculate
  $t_1 = g_1^w, t_2 = g_2^w$.
  calculate
  $c = H(ID, g_1, \Gamma_1, g_2, \Gamma_2, t_1, t_2)$
  calculate $r = w - cx$
  return $\eta = (c, r)$
end

Algorithm 8: Verification of proof η generated by Algorithm 7 given $ID, \Gamma_1, g_1, \Gamma_2, g_2$.

Input : $ID, \Gamma_1, g_1, \Gamma_2, g_2, \eta = (c, r)$
Output: success or failure
begin
  calculate
  $t_1 = g_1^r \Gamma_1^c, t_2 = g_2^r \Gamma_2^c$
  calculate
  $c_1 = H(ID, g_1, \Gamma_1, g_2, \Gamma_2, t_1, t_2)$
  if $c_1 = c$ **then**
    return success
  else
    return failure
  end
end

In this section, we present the NIZK proof algorithms that are required in Section 6.1.1. Algorithm 5 (resp. Algorithm 6) represents the prover algorithm (resp. verifier algorithm) for generation (resp. verification) of the NIZK proof $P_K\{token : \Gamma' = g^{token}\}$ required in the voting and tallying phase in Section 6.1.1 for an encrypted vote (U_i, V_i). Algorithm 7 (resp. Algorithm 8) represents the prover algorithm (resp. verifier algorithm) for generation (resp. verification) of the NIZK proof $P_K\{s : (\Gamma_1 = g_1^s) \wedge (\Gamma_2 = g_2^s)\}$ and $P_K\{s.prev\_hash : (\Gamma_3 = g_1^{s.prev\_hash}) \wedge (\Gamma_4 = g_2^{s.prev\_hash})\}$ required in the voting and tallying phase in Section 6.1.1.

Algorithm 9: A prover with identifier ID generates a proof of knowledge of a secret λ such that $(\Gamma' = \{\gamma'\}^\lambda) \wedge ((\Gamma_1'' = \{\gamma_1''\}^\lambda) \vee (\Gamma_2'' = \{\gamma_2''\}^\lambda))$.

Input : $ID, \Gamma', \gamma', (\Gamma_l'', \gamma_l'')_{l=1}^2, \lambda$ such that $\Gamma' = \{\gamma'\}^\lambda$ and $\Gamma_1'' = \{\gamma_1''\}^\lambda$
Output: $\Pi = P_K\{\lambda : (\Gamma' = \{\gamma'\}^\lambda) \wedge ((\Gamma_1'' = \{\gamma_1''\}^\lambda) \vee (\Gamma_2'' = \{\gamma_2''\}^\lambda))\}$
begin
  choose random $w, r_2, c_2 \in \mathbb{Z}_q$
  calculate $w_1 = w + r_2 + c_2\lambda$
  calculate

$$t_1 = \{\gamma'\}^{w_1}, t_{12} = \{\gamma_1''\}^w, t_{22} = \{\gamma_2''\}^{r_2}\{\Gamma_2''\}^{c_2} \quad \text{(B.1)}$$

  calculate
  $c = H(ID, \gamma', \Gamma', (\gamma_l'', \Gamma_l'')_{l=1}^2, t_1, (t_{l2})_{l=1}^2)$,

$$c_1 = c - c_2 \quad \text{(B.2)}$$

  calculate

$$r_1 = w - c_1\lambda \quad \text{(B.3)}$$

  return $\eta = (c_1, c_2, r_1, r_2)$
end

Appendix B

Security properties of zero-knowledge proof of prover Algorithm 3 and the verifier Algorithm 4.

We have proved the security properties of the proposed efficient NIZK proof for two candidates i.e. for 1-out-of −2 NIZK proof. The security proofs can be easily extended for n candidates i.e. for 1-out-of-n NIZK proof. Algorithm 9 (resp. Algorithm 10) is presented to prove (rep. verify) a proposition of the form $\varphi' \wedge (\varphi_1 \vee \varphi_2)$ for three assertions

Algorithm 10: Verification of proof Π generated by *Algorithm* 9 given $ID, \Gamma', \gamma', (\Gamma_l'', \gamma_l'')_{l=1}^2$. The discrete logarithmic relationships for the pairs (γ', γ_1'') and (γ', γ_2'') are unknown.

Input : $ID, \Gamma', \gamma', (\Gamma_l'', \gamma_l'')_{l=1}^2, \Pi = (c_1, c_2, r_1, r_2)$
Output: successful or failure
begin
  calculate
  $t_1 = \{\gamma'\}^{r_1+r_2}\{\Gamma'\}^{c_1+c_2}$, $t_{12} = \{\gamma_1''\}^{r_1}\{\Gamma_1''\}^{c_1}$ and
  $t_{22} = \{\gamma_2''\}^{r_2}\{\Gamma_2''\}^{c_2}$
  calculate
  $c' = H(ID, \gamma', \Gamma', (\gamma_l'', \Gamma_l'')_{l=1}^2, t_1, (t_{l2})_{l=1}^2)$
  if $c' = c_1 + c_2$ **then**
    return successful
  else
    return failure
  end
end

φ', φ_1 and φ_2, where the prover knows discrete logarithms for the pair (φ', φ_1). Here φ', φ_1 and φ_2 are assertions $(\Gamma' = \{\gamma'\}^{\lambda}), (\Gamma_1'' = \{\gamma_1''\}^{\lambda})$ and $(\Gamma_2'' = \{\gamma_2''\}^{\lambda})$ respectively. In order for prover P and verifier V to achieve the security properties, we must restrict the computational power of V or any attacker so that it is bounded by a polynomial in the size of common input. Clearly, without this restriction we need not talk about zero-knowledge since V of an unbounded computational power can find P's private input hidden behind common input. The discrete logarithm for the pairs (γ', γ_1'') and (γ', γ_2'') are unknown. Note that this assumption is also made in [4] for construction of 1-out-of-n NIZK. Otherwise, an attacker can find out the set corresponding to the witness i.e. the witness indistinguishability property will be lost.

Completeness:

By direct observation of the protocol, it is straightforward to see that the completeness property is preserved. This means that, if the prover generates (c_1, c_2, r_1, r_2) and follows the protocol instruction, the honest verifier will always accept it.

Soundness:

We need to find soundness error probability.

Let us assume that for the same commitment (t_1, t_{12}, t_{22}) with fixed r_2, c_2, w, two different response viz. (c_1, c_2, r_1, r_2) and $(c_1^1, c_2^1, r_1^1, r_2^1)$ are generated, where $c_1 \neq c_1^1$. Now we can compute a witness for λ i.e. $\lambda = (r_1 - r_1^1)/(c_1^1 - c_1)$. Therefore, the prover P knows the witness λ. Similarly, let us assume that for the same commitment (t_1, t_{12}, t_{22}) with fixed r_1, c_1, w, two different response viz. (c_1, c_2, r_1, r_2) and $(c_1^1, c_2^1, r_1^1, r_2^1)$ are generated, where $c_2 \neq c_2^1$. Now we can compute a witness for λ i.e. $\lambda = (r_2 - r_2^1)/(c_2^1 - c_2)$. Therefore, in this case also, the prover P knows the witness λ.

Suppose a prover, P^*, is a cheater, i.e., she does not know the correct discrete logarithm value for any of the pair (φ', φ_1) or (φ', φ_2).

For a commitment (t_1, t_{12}, t_{22}) she chooses in Eq. (B.1), the verifier is waiting for a response (c_1, c_2, r_1, r_2) such that

$$c_1 + c_2 = H(ID, \gamma', \Gamma', (\gamma_l'', \Gamma_l'')_{l=1}^2, \{\gamma'\}^{r_1+r_2}\{\Gamma'\}^{c_1+c_2}, \{\gamma_1''\}^{r_1}\{\Gamma_1''\}^{c_1}, \{\gamma_2''\}^{r_2}\{\Gamma_2''\}^{c_2}). \tag{B.1}$$

Since the prover, P^*, does not know the correct discrete logarithm, the best known strategy to compute such response is to guess (r_1, r_2) and any one of c_1 or c_2 first as follows:

1. picking random $r_1 \in_R \mathbb{Z}_q$ and $r_2 \in_R \mathbb{Z}_q$ uniformly;

2. picking either of c_1 or c_2 uniformly from the image space of the hash function H i.e. picking $c_2 \in_R Image(H)$ (assuming H has the same large output space as $\mathbb{Z}_q$)

3. computing

$c_1 = H(ID, \gamma', \Gamma', (\gamma_l'', \Gamma_l'')_{l=1}^2, \{\gamma'\}^{r_1+r_2}\{\Gamma'\}^{c_1+c_2}, \{\gamma_1''\}^{r_1}\{\Gamma_1''\}^{c_1}, \{\gamma_2''\}^{r_2}\{\Gamma_2''\}^{c_2}) - c_2$. (or $c_2 = H(ID, \gamma', \Gamma', (\gamma_l'', \Gamma_l'')_{l=1}^2, \{\gamma'\}^{r_1+r_2}\{\Gamma'\}^{c_1+c_2}, \{\gamma_1''\}^{r_1}\{\Gamma_1''\}^{c_1}, \{\gamma_2''\}^{r_2}\{\Gamma_2''\}^{c_2}) - c_1$, if he picks $c_1 \in_R Image(H)$ in step 2).

This is a well-known computationally hard problem to find such a c_1 since H is a random oracle and g^x is a one-way function. Now since she does not know the discrete logarithm corresponding to any of the pair (φ', φ_1) or (φ', φ_2) and H is a random oracle and g^x is a one-way function, the soundness error probability is $(1/2^n)$, where n is the security parameter $\log(q)$.

Zero-knowledgeness:

We will show that although the algorithm is not a full zero-knowledge, the algorithm does not reveal any information about P's private input λ. Note that the Schnorr signature scheme (NIZK) and the 1-out-of-n zero-knowledge proof described in [4] are also not a full zero-knowledge but they does not reveal any information about P's private input.

For a response (c_1, c_2, r_1, r_2) to be valid and accepted by the verifier V, they must satisfy the equation

$$c_1 = H(ID, \gamma', \Gamma', (\gamma_l'', \Gamma_l'')_{l=1}^2, \{\gamma'\}^{r_1+r_2}\{\Gamma'\}^{c_1+c_2}, \{\gamma_1''\}^{r_1}\{\Gamma_1''\}^{c_1}, \{\gamma_2''\}^{r_2}\{\Gamma_2''\}^{c_2}) - c_2. \tag{B.2}$$

Viewed by a third party, Eq. (B.2) means either of the following two cases:

1. the equation is constructed by P using her private input, hence P discloses that she has been in interaction with verifier V, or

2. an attacker or a verifier has successfully broken the random oracle hash function H of large output space $\mathbb{Z}_q$ and another one-way function g^x, because she has constructed Eq. (B.2).

This is a well-known hard problem since H is a random oracle and g^x is a one-way function.

Since an attacker or verifier is polynomially bounded, the third party will of course believe that (1) is the case. The response (c_1, c_2, r_1, r_2) is precisely a signature under Schnorr's signature scheme. Since only P could have issued such signature, the third party has made correct judgment.

A simulator cannot generate such response (c_1, c_2, r_1, r_2) without knowing the discrete logarithm corresponding to any of the pair (φ', φ_1) or (φ', φ_2).

Therefore, it is not a full zero-knowledge. However, we will show that the proof transcript (c_1, c_2, r_1, r_2) does not reveal any information about P's private input (discrete logarithm).

Let us consider Eq. (B.3) calculated by prover P i.e. $r_1 = w - c_1\lambda$. Here w is chosen uniformly from $\mathbb{Z}_q$ independent from all previous instances and λ is P's private input. After receiving a valid response (c_1, c_2, r_1, r_2), c_1 and r_1 are known to the verifier V and any attacker.

Note that c_2 and r_2 are chosen uniformly from $\mathbb{Z}_q$ independent from all previous instances. Let $\tilde{C}_2$ and $\tilde{R}_2$ denote random variables corresponding to c_2 and r_2 respectively.

c being the output of the hash function H is also uniformly distributed over the image space of H. Let us assume that the image space of H is $\mathbb{Z}_q$. Let $\tilde{C}$ denote the random variable corresponding to c.

We assume that there is a probability distribution for λ which determines a random variable $\tilde{\lambda}$. Let $\tilde{W}$ denote the random variable corresponding to w. The three random variables viz. $\tilde{C}_1$, $\tilde{W}$ and $\tilde{\lambda}$ determine a random variable $\tilde{R}$ over $\mathbb{Z}_q$ representing r_1, where $r_1 = w - c_1\lambda$.

Let $\tilde{t_1}$, $\tilde{t_{12}}$ and $\tilde{t_{22}}$ denote the random variables representing $t_1 = \{\gamma'\}^{w_1}$, $t_{12} = \{\gamma_1''\}^{w}$ and $t_{22} = \{\gamma_2''\}^{r_2}\{\Gamma_2''\}^{c_2}$ respectively, where $w_1 = w + r_2 + c_2\lambda$.

Now, since r_2 and c_2 are chosen uniformly and independently from $\mathbb{Z}_q$, the random variables $\tilde{C_2}$ and $\tilde{t_{22}}$ are also independent.

Also since r_2, c_2 and w are chosen uniformly and independently from $\mathbb{Z}_q$, the random variables $\tilde{C_2}$ and $\tilde{t_1}$ are also independent.

Hence the random variables $\tilde{C_2}$ and $\tilde{C}$ are also independent, where the random variable $\tilde{C}$ represents $c = H(ID, \gamma', \Gamma', (\gamma_l'', \Gamma_l'')_{l=1}^2, t_1, (t_{l2})_{l=1}^2)$.

Consider Eq. B.2 i.e. $c_1 = c - c_2$. Here c_2 is uniformly distributed and independent of c. Therefore, the random variable $\tilde{C_1}$ is also uniformly distributed and independent of $\tilde{C}$. This follows from following well-known result from probability theory.

Two random variables X_1 and X_2 are such that $(X_1, X_2) \in \mathbb{G} \times \mathbb{G}$, where $(\mathbb{G}, +)$ is a group. Given that (1) X_1 and X_2 are independent and (2) X_1 is uniform over $\mathbb{G}$. Let $X_3 = X_1 + X_2$, then (1) X_2 and X_3 are independent and (2) X_3 has uniform distribution over $\mathbb{G}$.

Since $\tilde{C_1}$ is independent of $\tilde{C}$, $\tilde{C_1}$ is also independent of $\tilde{W}$ and $\tilde{\lambda}$.

Therefore, three random variables $\tilde{W}$, $\tilde{C_1}$ and $\tilde{\lambda}$ are uniformly distributed over $\mathbb{Z}_q$ and independent of each other. Let n denote the security parameter $\log(q)$. Now, $Pr[\tilde{R} = r_1] = \Sigma_{w\in\mathbb{Z}_q}\Sigma_{c_1\in\mathbb{Z}_q} Pr[\tilde{W} = w]Pr[\tilde{C_1} = c_1]Pr[\tilde{\lambda} = (w - r_1)c_1^{-1}] = (1/2^n)\Sigma_{w\in\mathbb{Z}_q}\Sigma_{c_1\in\mathbb{Z}_q} Pr[\tilde{C_1} = c_1]Pr[\tilde{\lambda} = (w - r_1)c_1^{-1}] = (1/2^n)^2\Sigma_{w\in\mathbb{Z}_q}\Sigma_{c_1\in\mathbb{Z}_q} Pr[\tilde{\lambda} = (w - r_1)c_1^{-1}] = (1/2^n)^2\Sigma_{w\in\mathbb{Z}_q} 1 = (1/2^n)^2.(2^n) = (1/2^n)$.

$Pr[\tilde{R} = r_1|\tilde{\lambda} = \lambda] = \Sigma_{c_1\in\mathbb{Z}_q} Pr[\tilde{C_1} = c_1]Pr[\tilde{W} = r_1 + c_1\lambda] = (1/2^n)\Sigma_{c_1\in\mathbb{Z}_q} Pr[\tilde{W} = r_1 + c_1\lambda] = (1/2^n)$.

Therefore, $Pr[\tilde{\lambda} = \lambda|\tilde{R} = r_1] = (Pr[\tilde{\lambda} = \lambda]Pr[\tilde{R} = r_1|\tilde{\lambda} = \lambda]/Pr[\tilde{R} = r_1]) = (Pr[\tilde{\lambda} = \lambda].(1/2^n))/(1/2^n)$

$$= Pr[\tilde{\lambda} = \lambda]. \tag{B.3}$$

Therefore, the r_1 does not reveal any information about P's private input λ. r_1 forms a one-time pad (shift cipher) encryption of P's private input λ, which provides information-theoretic quality of security i.e. it has prefect secrecy.

Although we have used the same λ for constructing $t_1 = \{\gamma'\}^{w_1}$, where $w_1 = w + r_2 + c_2\lambda$. Due to hardness of the discrete logarithm problem, an attacker or a verifier cannot find w_1 from $\{\gamma'\}^{w_1}$. Also, the logarithmic relationship for the pairs (γ', γ_1''), (γ', γ_2'') are unknown. t_1 does not reveal any more information about P's private input λ than that has already been revealed by their common input $\Gamma' = \{\gamma'\}^{\lambda}$.

Therefore, the algorithm does not reveal any more information about P's private input λ than that has been revealed by their common inputs Γ', Γ_1'', and Γ_2''.

Witness Indistinguishability:

We have to show that the distribution of the conversation is independent of the qualified set A corresponding to P's private input λ. Since the prover generates a proof of knowledge of a secret λ such that $(\Gamma' = \{\gamma'\}^{\lambda}) \wedge ((\Gamma_1'' = \{\gamma_1''\}^{\lambda}) \vee (\Gamma_2'' = \{\gamma_2''\}^{\lambda}))$, a verifier and an attacker already know that the assertion $(\Gamma' = \{\gamma'\}^{\lambda})$ is in the qualified set A. Our aim is to prevent an attacker or a verifier from knowing that which one among $(\Gamma_1'' = \{\gamma_1''\}^{\lambda})$ and $(\Gamma_2'' = \{\gamma_2''\}^{\lambda})$ corresponds to the P's private input λ. Let $\varphi' = (\Gamma' = \{\gamma'\}^{\lambda})$ and $\varphi_l = (\Gamma_l'' = \{\gamma_l''\}^{\lambda})$, where $l = 1, 2$.

We use the same notation and random variables described in the previous section.

Since, w, c_2 and r_2 are chosen uniformly and independently from $\mathbb{Z}_q$ (i.e. $t_1 \in_R \mathbb{G}_q$ and $t_{12} \in_R \mathbb{G}_q$), $\tilde{t_1}$, $\tilde{t_{12}}$ are independent from the qualified assertion φ_1. Since the logarithmic relationships for the pairs (γ', γ_1'') and (γ', γ_2'') are unknown to attacker, $\tilde{t_1}$, $\tilde{t_{12}}$, $\tilde{t_{22}}$ are also independent from the qualified assertion φ_1.

Therefore, $\tilde{C}$ is also independent from the qualified assertion φ_1.

Since c_2 is chosen uniformly and independently from $\mathbb{Z}_q$, then the distribution of (c_1, c_2) is also independent of the qualified assertion φ_1.

Since c_1 is uniformly distributed and independent from w, from Eq. (B.3) we can conclude that $\tilde{R}$ is independent from $\tilde{W}$ and hence $\tilde{R}$ is independent of the assertion φ_1.

Therefore, the conversation (c_1, c_2, r_1, r_2) is independent of the qualified assertion φ_1 and hence the algorithm is witness indistinguishable. □

References

[1] Chaum Devid. Secret-ballot receipts: True voter-verifiable elections. IEEE Secur Priv 2004;2(1):38–47.

[2] Kohno Tadayosh, Stubblefield Adam, Rubin Aviel D, Wallach Dan S. Analysis of an electronic voting system. In: IEEE symposium on security & privacy. 2004.

[3] Hao F, Kreeger MN, Randell B, Clarke D, Shahandashti SF, Lee PH-J. Every vote counts: Ensuring integrity in large-scale electronic voting. USENIX J Elect Technol Syst 2014;2(3):1–25.

[4] Shahandashti Siamak F, Hao Feng. DRE-ip: A verifiable e-voting scheme without tallying authorities. In: The 21st European symposium on research in computer security. 2016.

[5] Benaloh JDC. Verifiable secret-ballot elections [Ph.D. thesis], Department of Computer Science, Yale University; 1987.

[6] Culnane C, Schneider SA. A peered bulletin board for robust use in verifiable voting systems. In: CSF 2014. p. 169–83.

[7] Chondros N, Zhang B, Zacharias T, Diamantopoulos P, Maneas S, Patsonakis C, et al. D-DEMOS: A distributed, end-to-end verifiable, internet voting system. In: ICDCS 2016. p. 711–20.

[8] Kiayias A, Kuldmaa A, Lipmaa H, Siim J, Zacharias T. On the security properties of e-voting bulletin boards. In: Catalano D, De Prisco R, editors. Security and cryptography for networks. Lecture notes in computer science, vol. 11035, Cham: Springer; 2018.

[9] Küsters Ralf, Truderung Tomasz, Vogt Andreas. Clash attacks on the verifiability of e-voting systems. In: Proceedings of the 2012 IEEE symposium on security and privacy. Washington, DC, USA: IEEE Computer Society; 2012, p. 395–409. http://dx.doi.org/10.1109/SP.2012.32.

[10] Rivest RL, Smith WD. Three voting protocols: ThreeBallot, VAV and twin. In: USENIX/ACCURATE electronic voting technology. 2007.

[11] Adida B. Helios: Web-based open-audit voting. In: USENIX security symp., vol. 17, 2008, p. 335–48.

[12] Ben-Nun J, Llewellyn M, Riva B, Rosen A, Ta-Shma A, Wikström D. A new implementation of a dual (paper and cryptographic) voting system. In: EVOTE2012: 5th int'l conf. on electronic voting. 2012. p. 315–29.

[13] Benaloh Josh, Lazarus Eric. The trash attack: An attack on verifiable voting systems and a simple mitigation. Technical report, technical report MSR-TR-2011-115, Microsoft; 2011.

[14] Nakamoto Satoshi. Bitcoin: A peer-to-peer electronic cash system. Consulted 2008;1(2012):28.

[15] Karafiloski Elena, Mishev Anastas. Blockchain solutions for big data challenges: A literature review. In: 17th IEEE international conference on smart technologies. p. 763–8.

[16] Lao Laphou, Li Zecheng, Hou Songlin, Xiao Bin, Guo Songtao, Yang Yuanyuan. A survey of IoT applications in blockchain systems: Architecture, consensus, and traffic modeling. ACM Comput Surv 2020;53(1). Article 18, 32 pages.

[17] Fan K, Ren Y, Wang Y, Li H, Yang Y. Blockchain-based efficient privacy preserving and data sharing scheme of content-centric network in 5G. IET Commun 2018;12(5):527–32.

[18] Kuo TT, Kim HE, Ohno-Machado L. Blockchain distributed ledger technologies for biomedical and health care applications. J Am Med Inform Assoc 2017;24(6):1211–20.

[19] Jabbar Karim, Bjørn Pernille. Infrastructural grind: Introducing blockchain technology in the shipping domain. In: Proceedings of the 2018 ACM conference on supporting groupwork. New York, NY, USA: Association for Computing Machinery; 2018, p. 297–308.

[20] Casino Fran, Dasaklis Thomas K, Patsakis Constantinos. A systematic literature review of blockchain-based applications: Current status, classification and open issues. Telemat Inform 2019;36:55–81.

[21] Juels A, Catalano D, Jakobsson M. Coercion-resistant electronic elections. In: Chaum D, Jakobsson M, Rivest RL, Ryan PYA, Benaloh J, Kutylowski M, Adida B, editors. Towards trustworthy elections, new directions in electronic voting. Berlin, Germany: Springer; 2010, p. 37–63.

[22] Nandakumar K, Jain AK, Pankanti S. Fingerprint-based fuzzy vault: Implementation and performance. IEEE Trans Inf Forensics Secur 2007;2(4):744–57. http://dx.doi.org/10.1109/TIFS.2007.908165.

[23] Lin Y, Zhang P. Blockchain-based complete self-tallying e-voting protocol. In: 2019 Asia-Pacific signal and information processing association annual summit and conference. Lanzhou, China. 2019. p. 47–52. http://dx.doi.org/10.1109/APSIPAASC47483.2019.9023220.

[24] Neff CA. Practical high certainty intent verification for encrypted votes. 2004, Available from http://citeseer.ist.psu.
[25] Ryan P, Bismark D, Heather J, Schneider S, Xia Z. Prêt à Voter: A voter-verifiable voting system. IEEE Trans Inf Forensics Secur 2009;4(4):662–73.
[26] Bell S, Benaloh J, Byrne MD, DeBeauvoir D, Eakin B, Fisher G, et al. STAR-Vote: A secure, transparent, auditable, and reliable voting system. USENIX J Elect Technol Syst 2013;1(1):18–37.
[27] Fisher K, Carback R, Sherman AT. Punchscan: Introduction and system definition of a high-integrity election system. In: workshop on trustworthy elections. 2006.
[28] Adida B, Rivest RL. Scratch & vote: Self-contained paper-based cryptographic voting. In WEPS06: Proceedings of the 5th ACM workshop on privacy in electronic society. New York: 2006.
[29] Chaum D, Carback R, Clark J, Essex A, Popoveniuc S, Rivest R, et al. Scantegrity II: End-to-end verifiability by voters of optical scan elections through confirmation codes. IEEE Trans Inf Forensics Secur 2009;4(4):611–27.
[30] Bohli J-M, Müller-Quade J, Röhrich S. Bingo voting: Secure and coercion-free voting using a trusted random number generator. In: E-voting and identity. Springer; 2007, p. 111–24.
[31] Hao Feng, Ryan Peter YA, editors. Real-world electronic voting: Design, analysis and deployment. Series in security, privacy and trust, CRC Press; 2016.
[32] Culnane C, Ryan PYA, Schneider S, Teague V. VVote: A verifiable voting system. ACM Trans Inf Syst Secur 2015;18(1):3:1–30.
[33] Carback R, Chaum D, Clark J, Conway J, Essex A, Herrnson P, et al. Scantegrity II municipal election at Takoma Park: The first E2E binding governmental election with ballot privacy. In: USENIX security symp. 2010. p. 291–306.
[34] Zhao Z, Chan THH. How to vote privately using bitcoin. In: International conference on information and communications security. Springer; 2015, p. 82–96.
[35] Tarasov P, Tewari H. Internet voting using zcash. Cryptology ePrint archive, report 2017/585, 2017, http://eprint.iacr.org/2017/585.
[36] McCorry P, Shahandashti S, Hao F. A smart contract for boardroom voting with maximum voter privacy. Cryptology ePrint archive, report 2017/110, 2017, https://eprint.iacr.org/2017/110.
[37] Seifelnasr M, Galal HS, Youssef AM. Scalable open-vote network on ethereum. In: Bernhard M, et al., editors. Financial cryptography and data security. Lecture notes in computer science, vol. 12063, 2020.
[38] Tivi voting. 2016, https://tivi.io/. [Accessed 25 October 2018].
[39] Follow my vote. 2018, https://followmyvote.com/. [Accessed 25 October 2018].
[40] Hertig A. The first bitcoin voting machine is on its way. 2015, Available: https://www.vice.com/en/article/3da8e5/the-first-bitcoin-voting-machine-is-on-its-way. [Accessed 20 January 2021].
[41] Higgins S. Abu Dhabi stock exchange launches blockchain voting. 2016, Available: https://www.coindesk.com/abu-dhabi-exchange-blockchain-voting. [Accessed 20 January 2021].
[42] Boucher P. What if blockchain technology revolutionised voting. 2016, Available: http://www.europarl.europa.eu/RegData/etudes/ATAG/2016/581918/EPRS_ATA(2016)581918_EN.pdf. [Accessed January 2021].
[43] Van der Elst C, Lafarre A. Blockchain and smart contracting for the shareholder community. Eur Bus Organ Law Revice 2019;20(1):111–37.
[44] Gaudry P, Golovnev A. Breaking the encryption scheme of the Moscow internet voting system. In: Bonneau J, Heninger N, editors. Financial cryptography and data security. Lecture notes in computer science, vol. 12059, Cham: Springer; 2020.
[45] Kremer S, Ryan M, Smyth B. Election verifiability in electronic voting protocols. In: Gritzalis D, Preneel B, Theoharidou M, editors. Computer security - ESORICS 2010. Lecture notes in computer science, vol. 6345, Berlin, Heidelberg: Springer; 2010.
[46] Camenisch J, Stadler M. Efficient group signature schemes for large groups. In: Crypto'97. LNCS, vol. 1294, Springer; 1997, p. 410–24.
[47] Diffie W, Hellman ME. New directions in cryptography. IEEE Trans Inform Theory 1976;22(6):644–54.
[48] Stinson Douglas. Cryptography: Theory and practice. 2nd ed.. CRC/C & H; 2002.
[49] Tomko George J, Soutar Colin, Schmidt Gregory J. Fingerprint controlled public key cryptographic system. 1996. US Patent 5, 541, 994.
[50] Jain Anil K, Nandakumar Karthik, Nagar Abhishek. Biometric template security. EURASIP J Adv Signal Process 2008. http://dx.doi.org/10.1155/2008/579416, Article 113, 17 pages.
[51] Tuyls PT, Skoric B, Kevenaar TAM, editors. Security with noisy data: On private biometrics, secure key storage and anti-counterfeiting. Germany: Springer; 2007.
[52] Boulgouris NV, Plataniotis Konstantinos N, Micheli-Tzanakou Evangelia. Biometric encryption: The new breed of untraceable biometrics. In: Biometrics: Theory, methods, and applications. IEEE; 2010, p. 655–718. http://dx.doi.org/10.1002/9780470522356.ch26.
[53] Cavoukian Ann, Stoianov Alex. Biometric encryption chapter from the encyclopedia of biometrics the following is a chapter on biometric encryption excerpted from the springer encyclopedia of biometrics. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.454.8371.
[54] Goldwasser S, Micali S, Rackof C. The knowledge complexity of interactive proof systems. SIAM J Comput 1989;18(1):186–208.
[55] Bellare M, Goldreich O. On defining proofs of knowledge. In: Brickell EF, editor. Crypto'92. LNCS, vol. 740, Springer; 1993, p. 390–420.
[56] Schnorr C-P. Efficient signature generation by smart cards. J Cryptology 1991;4(3):161–74.
[57] Cramer R, Damgård I, Schoenmakers B. Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt Y, editor. Crypto'94. LNCS, vol. 839, Springer; 1994, p. 174–87.
[58] Fiat A, Shamir A. How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko AM, editor. Crypto'86. LNCS, vol. 263, Springer; 1987, p. 186–94.
[59] Bellare M, Rogaway P. Random oracles are practical: A paradigm for designing efficient protocols. In: ACM CCS'93. ACM; 1993, p. 62–73.
[60] Wood G. Ethereum: A secure decentralised generalised transaction ledger. 2014, Available: https://gavwood.com/paper.pdf.
[61] Back Adam. Hashcash - A denial of service counter-measure. 2015, Available: http://www.hashcash.org/papers/hashcash.pdf. [Accessed 15 December 2020]. 2002.
[62] Chaum Devid. Untraceable electronic mail, return addresses, and digital pseudonyms. Commun ACM 1981;24(2):84–8.
[63] Sako Kazue, Kilian Joe. Receipt-free mix-type voting scheme - A practical solution to the implementation of a voting booth. In: Advances in Cryptology - EUROCRYPT '95, international conference on the theory and application of cryptographic techniques. Lecture notes in computer science, vol. 921, New York, NY, USA: Association for Computing Machinery; 1995, p. 297–308.
[64] Andrew Neff C. A verifiable secret shuffle and its application to e-voting. In: Proceedings of the 8th ACM conference on computer and communications security. New York, NY, USA: Association for Computing Machinery; 2001, p. 116–25. http://dx.doi.org/10.1145/501983.502000.
[65] Yao Andrew Chi-Chih. How to generate and exchange secrets (extended abstract). In: 27th annual symposium on foundations of computer science. IEEE Computer Society; 1986, p. 162–7.
[66] Goldreich O, Micali S, Wigderson A. How to play any mental game - A completeness theorem for protocols with honest majority. In: 19th ACM symposium on the theory of computing. 1987. p. 218–29.
[67] Ben-Or M, Goldwasser S, Wigderson A. Completeness theorems for non cryptographic fault tolerant distributed computation. In: 20th STOC. 1988. p. 1–9.
[68] Goldreich O. Secure multiparty computation (working draft), 1998, Available: http://www.wisdom.weizmann.ac.il/~oded/pp.html.
[69] Benaloh J. Ballot casting assurance via voter-initiated poll station auditing. In: USENIX workshop on accurate e-voting technology. 2007. p. 14.
[70] Adida Ben, de Marneffe Olivier, Pereira Olivier, Quisquater Jean-Jaques. Electing a university president using open-audit voting: Analysis of real-world use of helios. In: USENIX/ACCURATE electronic voting technology. 2009.
[71] Sandler Daniel, Wallach Dan S. Casting votes in the auditorium. In: USENIX/ACCURATE electronic voting technology workshop. 2007.
[72] Sandler D, Derr K, Wallach DS. VoteBox: A tamper-evident, verifiable electronic voting system. In: USENIX security symp, vol. 4, 2008, p. 87.
[73] Gennaro R, Ishai Y, Kushilevitz E, Rabin T. On 2-round secure multiparty computation. In: Proc. 22nd annu. int. cryptol. conf. adv. cryptol. 2002. p. 178–93.
[74] Palngipang JF, Ting ML, Atienza R. BBCast: Cloud-based interactive public bulletin board. In: 9th international conference on next generation mobile applications, services and technologies. Cambridge: 2015. p. 35–40.
[75] Benet Juan. IPFS - Content addressed, versioned, P2P file system. 2014, Available: https://arxiv.org/abs/1407.3561.
[76] Feng H, Llorca J, Tulino AM, Molisch AF. Optimal dynamic cloud network control. IEEE/ACM Trans Netw 2018;26(5):2118–31.
[77] Salah K. A queueing model to achieve proper elasticity for cloud cluster jobs. In: IEEE sixth international conference on cloud computing. Santa Clara, CA: 2013. p. 755–61.
[78] Hao F, Clarke D, Zorzo A. Deleting secret data with public verifiability. IEEE Trans Dependable Secure Comput 2015;PP(99):1.
[79] Benaloh J, Tuinstra D. Receipt-free secret-ballot elections. In: 26th ACM Symp. on theory of computing. ACM; 1994, p. 544–53.
[80] Menezes AJ, van Oorschot PC, Vanstone SA. Handbook of applied cryptography. CRC press; 1996.

Somnath Panja received the M.Tech degree in Computer Science from Indian Statistical Institute, Kolkata, India, in 2010, and the M.Sc degree in Mathematics from the Indian Institute of Technology Madras, India, in 2008. He is currently a Senior Research Fellow at the Applied Statistics Unit, Indian Statistical Institute, Kolkata. His primary research interests include cryptography and security, secret sharing, e-voting, blockchain, secure multi-party computation and their applications.

Bimal Roy received the Ph.D. degree in combinatorics from the University of Waterloo, Waterloo, Canada, in 1982. He is currently a Professor and the Head of R C Bose Centre for Cryptology and Security, Indian Statistical Institute, Kolkata, India. He is also the Founder and General-Secretary, Cryptology Research Society of India. He served as the Director of Indian Statistical Institute, Kolkata, from 2010 to 2015. Over the past 40 years, he has published many research papers on the subject of Cryptography. His primary research interests include Cryptology, Data obfuscation, Design of secure Electronic voting machine, Sensor Networks, Combinatorics, Design of Experiments, and Optimization.