End-to-End Verifiable Cumulative Voting without Tallying Authorities

Samiran Bag[1], Muhammad Ajmal Azad[2], and Feng Hao[1]

[1] Department of Computer Science
University of Warwick
{samiran.bag,feng.hao}@warwick.ac.uk
[2] Department of Computer Science
University of Derby
m.azad@derby.ac.uk

Abstract. In this paper, we propose the first end-to-end (E2E) verifiable e-voting system for cumulative voting without requiring any tallying authorities. Cumulative voting is an electoral system, heavily used in corporate governance as practised in several US states, and in participatory budgeting as seen in many European cities where local residents decide how to spend a portion of the local government's budget through voting. Traditionally, cumulative voting is done with pen and paper, but the manual counting process is time consuming and costly, especially when such voting events occur frequently. Many systems have changed to use electronic voting, but without considering the security aspects of this change. To our knowledge, none of the existing e-voting systems implemented for cumulative voting are end-to-end verifiable; if there is any bug or tempering at the tallying software, the tally would be inadvertently modified without any voter noticing this. Although there are existing voting systems (e.g., mix-net based) that could be adapted to support cumulative voting with E2E verifiability, they generally require a set of tallying authorities, which can lead to substantial complexity of finding and managing such authorities in practice. We address this issue by adopting novel cryptographic techniques to achieve E2E verifiability for cumulative voting, but without involving any tallying authorities. We formally define a model to prove the security of our system, and present the efficiency analysis to show that our proposed solution is feasible for practical use.

Keywords: end-to-end verifiability, verifiable e-voting, cumulative voting, provable security, receipt-freeness.

1 Introduction

Cumulative voting is an electoral system in which a voter is given a number of votes to vote for a number of chosen candidates. A voter may give all their votes to one candidate or divide them among candidates. Typically, each voter

is assigned the number of votes equal to the number of seats to be filled in the election. For example, if the election is to be held for five seats, then each voter is allocated with five votes. If a voter particularly favors a candidate, they may give all votes to that candidate. Alternatively, they may spread votes among candidates according to their preference. The winners are decided based on who have got the five most votes. Cumulative voting is frequently used in corporate governance as seen in several US states [34]. It is also commonly used among European cities such as Barcelona and Gdansk [29] for participatory budgeting, where local residents decide how to spend a portion of the city council budget by voting against several selected projects.

Traditionally, cumulative voting is done with pen and paper. Voters fill in paper ballots and hand over the completed ballots to a returning officer or post them by mail. When all completed ballots are collected, they are counted by trustworthy tallying authorities (TAs), and winners are announced in the end. With paper ballot, voters can verify the election process to some extent – more specifically they can verify that their votes are cast as intended (since they fill in the ballot themselves), but not how the cast votes will be recorded and tallied after they are put into a ballot box or posted out.

Besides the limited extent of voter verifiability, paper ballots have several practical limitations. First of all, casting paper ballots is not easy to handle by ordinary voters (particularly those with disabilities). Even when postal voting is allowed, voters will still need to post out ballots, by physically visiting a letter box or a postal office. This may be seen as a small step, but it can present a significant barrier when voters are expected to do this frequently. Furthermore, the counting process of physical ballots can be time consuming and costly. For major elections that happen infrequently (say once every four years), public observation of the counting process can be arranged by inviting representatives from different parties to be present at the tallying venue as independent observers. But for more frequent and smaller scale elections, the counting is often outsourced to external companies (e.g., Electoral Reform Society in the UK[3]) who act as a trusted third party.

To improve efficiency and accessibility, voting using electronic means has been actively explored by researchers in the field. Over the last three decades, many electronic voting schemes have been proposed for the Internet voting and polling station voting [2,3,5,16,18,22,24,27,33,35–38]. Electronic voting systems can be classified into three types: 1) systems based on blind signatures and anonymous channels [17], 2) based on mixing among servers [3,16,33,35], and 3) based on the homomorphic encryption [2, 5, 27, 38]. These schemes have proposed to achieve the fundamental properties of an electronic voting process: a) voter privacy, i.e., the value of each individual vote is kept secret and unlinkable to any particular voter, b) individual verifiability, i.e., each voter is able to verify that their vote is cast as intended and recorded as cast; c) universal verifiability, i.e. anyone is able to verify that all votes are tallied as recorded; d) receipt-freeness, i.e., voters would not be able to prove to any third party that they voted for a particular

[3] `https://www.electoral-reform.org.uk/`

candidate; e) robustness, i.e., the election process is not stopped even in the presence of malicious voters. These schemes are designed to ensure voter privacy and tallying integrity for the particular types of an election.

The design of an electronic voting system for cumulative voting should satisfy the following verifiability requirements.

Cast as intended: Every voter is able to verify that their cast vote captures their true intention.

Recorded as cast: Every voter is able to verify that their cast vote is recorded by the system and hence must be included into the tallying process.

Tallied as recorded: Every one is able to verify that all the recorded votes are tallied correctly in the tallying process.

A voting system that satisfies all the three above-mentioned requirements is said to be end-to-end (E2E) verifiable. In the meantime, an E2E voting system should also preserve the voter privacy – a coercer must not be able to learn a voter's vote and likewise the voter must not be able to prove to any third party whom they have voted for. Starting from Chaum's seminal paper on E2E verifiable voting [18], many E2E voting systems have been proposed. Most of the existing systems are dependent on a set of tallying authorities (TAs), who are supposedly trustworthy individuals with cryptographic and computing expertise tasked to perform complex cryptographic operations. These TAs should be selected from different parties with conflicting interest, so collusion among them is unlikely. They should also be subject to a cryptographic threshold control so that it takes compromising over a threshold of authorities to comprise the system. The threshold should be set sufficiently high to make collusion difficult, but also sufficiently low so that the decryption and tallying process is fault-tolerant. Although the use of trustworthy tallying authorities in e-voting is motivated by the similar role of tallying authorities required in the physical ballot counting process, implementing and managing such authorities in the e-voting context has proved to be particularly difficult [2]. This is one major obstacle that has prevented E2E verifiable voting systems from being widely used in practice [28].

Hao et al. [27] proposed the first E2E verifiable e-voting system without any tallying authorities called "DRE-i" (DRE with integrity). Their protocol is designed for the DRE (Direct Recording Electronic)-based election but the same protocol can also be implemented for remote internet voting [26]. In DRE-i, the encrypted ballot is constructed using a variant of the ElGamal encryption scheme with dynamically constructed public keys, such that after the election multiplying all ciphertexts will cancel out all the random factors introduced in the encryption process, hence allowing anyone to verify the tally of an election without needing any tallying authorities. The authors call such a TA-free system as "self-enforcing e-voting". Since DRE-i works by pre-computing the encrypted ballots before an election, the pre-computed ballots will naturally need to be stored securely, otherwise the secrecy of the votes will be affected. Shahandashti and Hao [39] addressed this issue by proposing an alternative real-time computation strategy, based on which they designed a new E2E verifiable voting system

without tallying authorities, called "DRE-ip" (DRE-i with enhanced privacy). This scheme does not compute the ciphertexts in advance; instead it creates the encrypted ballot on the fly. DRE-ip achieves E2E verifiability without tallying authorities, but at the same time provides a stronger privacy guarantee than DRE-i. However, both the DRE-i and DRE-ip systems are designed to support only plurality or approval voting and they do not support more complex ranking-based electoral systems such as cumulative voting.

In this paper, we proposed an E2E verifiable voting scheme for cumulative voting without involving any tallying authorities. Our design is inspired by the real-time computation strategy used in DRE-ip but we significantly modify it to support a cumulative voting scheme with the inherent properties of privacy, verifiability and correctness. Our approach only reveals the total score that each candidate gets without showing any distribution of votes within a ballot, hence effectively preventing Italian attacks (in which a coercer may force a voter to vote in an obscure pattern and check if the voter has indeed followed the instruction by searching for such a pattern in the data published during the tallying process; see [35]). When the DRE machine is completely compromised, our protocol still guarantees the tallying integrity, and meanwhile limits the attacker to learn only the partial tally at the time of compromise, which is minimal information leakage as one may hope for.

The contributions of this paper can be summarized as below.

- We propose DRE-CV, a new E2E verifiable e-voting system without TA that implements the cumulative voting scheme.
- We propose a new formal definition of an end-to-end verifiable e-voting system without tallying authority, and a model that defines the security properties of such an e-voting system.
- We show that our proposed cumulative voting system is provable secure under the proposed formal model.
- Lastly, we analyze the efficiency of our proposed system and show that it is feasible for practical use.

2 Preliminaries

2.1 Notations

Let G be a finite algebraic group and $A, B, g \in G$. Let $a = \log_g A$ and $b = \log_g B$. We denote $DH_g(A, B) = g^{ab}$.

2.2 Cumulative Voting

In cumulative voting, a voter can vote for more than one candidates. Besides, the voter gets the freedom to split her vote and distribute it among her chosen candidates. There are many variants of cumulative voting system. In this paper, we consider 'fractional ballot' voting scheme. Here, the voter receives a set of

points which she gives away to the candidates the way she wishes to. She can transfer all her points to a single candidate or she can distribute in any premeditated manner. Upon the completion of the election, the points assigned to a candidate by different voters are summed up, and the winners are chosen on the basis of the total points obtained by the candidates.

2.3 NIZK Proofs

An efficient-prover non-interactive zero-knowledge proof [25] for the relation $R \in L$ consists of three probabilistic polynomial time algorithms $\Gamma = (K, P, V)$ as described below:

- K is the CRS generating algorithm. It takes as input a λ at returns a common reference string, i.e. $\sigma \leftarrow K(1^\lambda)$.
- P is the prover algorithm. It takes as input the CRS σ a statement $x \in L$ and a witness w such that $R(x, w) = True$, and returns a proof π. That is, $\pi \leftarrow P(\sigma, x, w)$.
- V is the verifier algorithm. It takes as input the CRS σ, the statement x, the proof π, and returns $v \in \{0, 1\}$. That is, $v \leftarrow V(\sigma, x, \pi)$.

Any non-interactive zero knowledge proof system (K, P, V) must satisfy the following properties.

- *Completeness:* The NIZK proof system (K, P, V) is complete if

  $$Pr\left[\sigma \leftarrow K(1^\lambda); \pi \leftarrow P(\sigma, x, w), V(\sigma, x, \pi) = 1 \wedge R(x, w) = True\right] \geq 1 - negl(\lambda)$$

- *Soundness:* The NIZK proof system (K, P, V) is sound if

  $$Pr\left[\sigma \leftarrow K(1^\lambda); (x, \pi) \leftarrow \mathcal{A}(\sigma); V(\sigma, x, \pi) = 1 \wedge x \notin L\right] \leq negl(\lambda)$$

- *Zero Knowledge:* The NIZK proof system is zero knowledge if there exists a simulator $S = (S_1, S_2)$ such that for all probabilistic polynomial time adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$,

  $$Adv^{\Gamma}_{\mathcal{A},UZK}(\lambda) = \left| Pr\left[Expt^{REAL}_{\mathcal{A}}(\lambda) = 1\right] - Pr\left[Exp^{SIM}_{\mathcal{A},S}(\lambda) = 1\right] \right| \leq negl(\lambda)$$

$Exp^{REAL}_{\mathcal{A}}(\lambda)$	$Exp^{SIM}_{\mathcal{A},S}(\lambda)$
$\sigma \leftarrow K(1^\lambda)$	$(\sigma, \kappa) \leftarrow S_1(1^\lambda)$
$(x, w, \tau) \leftarrow \mathcal{A}_1(\sigma)$	$(x, w, \tau) \leftarrow \mathcal{A}_1(\sigma)$
$\pi \leftarrow P(\sigma, x, w)$	$\pi \leftarrow S_2(x, \kappa)$
Return $\mathcal{A}_2(\pi, \tau)$	Return $\mathcal{A}_2(\pi, \tau)$

2.4 Assumption

We will describe our system in the context of polling station voting in a supervised environment using touch-screen Direct Record Electronic (DRE) machines. We assume that the DRE machines used in our scheme are securely connected to an append-only database called the Bulletin Board [27,28,38]. Only the legitimate DRE machines can make posts on the Bulletin Board, whereas everyone else can only read from it. Every datum posted on the Bulletin Board by the DRE machine is digitally signed to prove authenticity of the published data. An adversary who does not have access to the signature-key cannot add entries on the Bulletin Board. Deletion of entries by the adversary can be detected if the entries on the Bulletin Board are hash-linked. Bulletin Boards have been widely used in the context of e-voting. There are many ways to implement a Bulletin Board: one way is to use mirror websites. If the adversary wants to change the content, she will have to ensure that the change is reflected on all sites, which is difficult. Another way of realizing a Bulletin Board is to use blockchain. The blockchain is secured by the miners and it is computationally infeasible for attackers to alter the established content stored on the blockchain. However, in this paper, we do not discuss security against malicious Bulletin Board. The reader may find discussion on this topic in [21].

3 Formal Definition

In this section, we formally define a DRE based E2E verifiable e-voting scheme. There are a few formal definitions on verifiable e-voting system. They will be discussed in section 9.

A DRE based E2E verifiable e-voting scheme consists of the following probabilistic algorithms. Let us assume f be the function that computes the tally of an election given the votes $v_1, v_2, \ldots, v_n$. The tally is denoted as $T = f(v_1, v_2, \ldots, v_n)$. Note that the tally does not depend upon the order in which the voters come. Hence, if $(v'_1, v'_2, \ldots, v'_n)$ be a permutation of $(v_1, v_2, \ldots, v_n)$, then $f(v'_1, v'_2, \ldots, v'_n) = f(v_1, v_2, \ldots, v_n)$.

Setup It is a probabilistic polynomial time algorithm that takes as input, the security parameter λ, and an election specific parameter α, and outputs the public parameters prm, the maximum number of votes $n_{\max}$, the number of candidates c, the set of all acceptable votes $\mathcal{V}$, the tally space $\mathcal{T}$, the trapdoor space $\mathfrak{T}$, the state space $\mathcal{S}$, the ballot space $\mathfrak{B}$, and the initial state st_{init}. In other words;

$$(prm, n_{\max}, c, \mathcal{V}, \mathcal{T}, \mathfrak{T}, \mathcal{S}, \mathfrak{B}, st_{init}) \leftarrow Setup(\lambda, \alpha)$$

BallotGen It is a probabilistic polynomial time algorithm that takes as input the public parameter prm, an initial state st_{in}, a vote $v \in \mathcal{V}$, and outputs a randomized ballot $b \in \mathfrak{B}$, a NIZK proof π, a trapdoor τ, and an updated state

st_{out}. (Details about the NIZK proof construction can be found in Appendix.) We may write this procedure as following

$$(b, \pi, \tau, st_{out}) \leftarrow BallotGen(prm, v, st_{in})$$

VerifyAudit It is a deterministic polynomial time algorithm that takes as input the public parameter prm, a randomized ballot $b \in \mathfrak{B}$, a unique trapdoor τ, and returns a vote $v \in \mathcal{V}$.

$$(\mathcal{V} \cup \{\perp\}) \leftarrow VerifyAudit(prm, b, \tau)$$

For any $b \in \mathfrak{B}, \tau, \tau' \in \mathfrak{T}, VerifyAudit(prm, b, \tau) \neq \perp$ and $VerifyAudit(prm, b, \tau') \neq \perp$, then $\tau = \tau'$. This property is called the uniqueness property of trapdoors and is formally defined later.

VerifyBallot It is a deterministic polynomial time algorithm that takes as input the public parameter prm, a ballot $b \in \mathfrak{B}$, and the associated NIZK proof π, and the vote space $\mathcal{V}$ and returns either $True$ or $False$. That is:

$$\{True, False\} \leftarrow VerifyBallot(prm, b, \pi, \mathcal{V})$$

The $VerifyBallot(prm, b, \pi, \mathcal{V})$ algorithm returns $True$ with overwhelming probability, if $\exists \tau \in \mathfrak{T}$, such that $VerifyAudit(b, \tau) \in \mathcal{V}$.

Tally It is a deterministic polynomial time algorithm that takes as input the public parameter prm a set of ballots $\mathbf{B} = (b_1, b_2, \ldots, b_r)$, where $r \in [1, n_{\max}]$, an initial state st_1 a final state st_f, and returns either a tally $T \in \mathcal{T}$ or $\perp$. In mathematical notation:

$$(\mathcal{T} \cup \{\perp\}) \leftarrow Tally(prm, B, st_f, st_1)$$

3.1 Correctness

The above DRE based E2E verifiable e-voting scheme is correct if the following probability is overwhelmingly high.

$$Pr\left[\begin{array}{c} (prm, n_{\max}, c, \mathcal{V}, \mathcal{T}, \mathfrak{T}, \mathcal{S}, \mathfrak{B}, st_{init}) \leftarrow Setup(\lambda, \alpha) \\ \forall n \leq n_{\max}, \forall v_1, v_2, \ldots, v_n \in \mathcal{V} \\ st_1 \leftarrow st_{init}, i \leftarrow 1, K \leftarrow 0 \\ \text{WHILE } (i \leq n) \\ (b_i, \pi_i, \tau_i, st_{i+1}) \leftarrow BallotGen(prm, v_i, st_i) \\ \text{IF } (VerifyBallot(prm, b_i, \pi_i, \mathcal{V}) = False \vee v_i \neq VerifyAudit(prm, b_i, \tau_i)) \\ \text{Then } K \leftarrow K + 1 \\ i \leftarrow i + 1 \\ \text{END WHILE} \\ B \leftarrow (b_1, b_2, \ldots, b_n) \\ T \leftarrow Tally(prm, B, st_{n+1}, st_1) \\ (K = 0) \wedge (T = f(v_1, v_2, \ldots, v_n)) \end{array}\right] \quad (1)$$

The correctness property implies that given any number n (polynomial in λ) of voters,

- If $BallotGen(prm, v, st)$ algorithm returns (b, π, τ, st'), then $VerifyBallot(prm, b, \pi, \mathcal{V})$ must return $True$
- If $BallotGen(prm, v, st)$ algorithm returns (b, π, τ, st'), then $VerifyAudit(prm, b, \tau)$ must return v
- If $(b_i, \pi_i, \tau_i, st_{i+1}) \leftarrow BallotGen(prm, v_i, st_i)$, for $i \in [1, n]$, then the $Tally(prm, B, st_{n+1}, st_1)$ function must return the tally of all votes, i.e. $f(v_1, v_2, \ldots, v_n)$, where B is the set of all ballots.

Uniqueness of the Trapdoor The trapdoor associated with a DRE based e-voting scheme $(Setup, BallotGen, VerifyBallot, VerifyAudit, Tally)$ is said to be bound to a ballot if for all probabilistic polynomial time algorithm $\mathcal{A}$,

$$Pr\left[Exp_{\mathcal{A},TBND}(\lambda) = 1\right] = 0$$

$Exp_{\mathcal{A},TBND}(\lambda)$
$(prm, n_{\max}, c, \mathcal{V}, \mathcal{T}, \mathfrak{T}, \mathcal{S}, \mathfrak{B}, st_{init}) \leftarrow Setup(\lambda, \alpha)$
$Aux = (n_{\max}, c, \mathcal{V}, \mathcal{T}, \mathfrak{T}, \mathcal{S}, \mathfrak{B}, st_{init})$
$(b, \tau, \tau') \leftarrow \mathcal{A}^{BallotGen, VerifyBallot, VerifyAudit, Tally}(prm, Aux)$
$v = VerifyAudit(b, \tau)$
$v' = VerifyAudit(b, \tau')$
Return $(v \neq \perp) \wedge (v' \neq \perp) \wedge (v \neq v')$

3.2 Verifiability

In this section, we provide the definitions of verifiability of a DRE based e-voting scheme. There exist a few definitions on verifiability. These will be discussed in section 9. While the general requirements for end-to-end verifiability are the same, our formalization of verifiability differs from previous works in that we do not require any trustees holding a secret key. In our setting, the DRE machine changes its state every time it generates a ballot. This state parameter serves as an input to the ballot generation algorithm which updates the state. On the contrary, in other existing definitions, the ballot generation function does not take as input any state function; rather the ballot generation algorithm takes the public key of the tallying authority and the vote as inputs. In our DRE based voting system, there is no public key of any tallying authority, and the verifiability of the election depends upon the final state of the DRE machine which is posted on the bulletin board. Due to this reason, we introduce a notion called 'auditability'. This notion underpins the "individual verifiability" proposed in the literature and at the same time allows protection against adversarial manipulation of the state parameter in order to deceive voters. Thus, the auditability notion attempts to assess the robustness of the voting systems in terms of individual verifiability when the adversary controls the DRE system.

Universal Verifiability A DRE based e-voting scheme $(Setup, BallotGen, VerifyBallot, VerifyAudit, Tally)$ is said to be universally verifiable if for all PPT adversary $\mathcal{A}$, $Adv_{\mathcal{A},SNDN}(\lambda)$ is negligible, where

$$Adv_{\mathcal{A},SNDN}(\lambda) = Pr[Exp_{\mathcal{A},SNDN}(\lambda) = 1]$$

$Exp_{\mathcal{A},SNDN}(\lambda)$
$(prm, n_{\max}, c, \mathcal{V}, \mathcal{T}, \mathfrak{T}, \mathcal{S}, \mathfrak{B}, st_{init}) \leftarrow Setup(\lambda, \alpha)$
$Aux = (n_{\max}, c, \mathcal{V}, \mathcal{T}, \mathfrak{T}, \mathcal{S}, \mathfrak{B}, st_{init})$
$(b, \pi) \leftarrow \mathcal{A}^{BallotGen, VerifyBallot, VerifyAudit, Tally}(prm, Aux)$
$(VerifyBallot(b, \pi) = True) \wedge (\forall \tau \in \mathfrak{T} : VerifyAudit(b, \tau) = \perp)$

Auditability A DRE based e-voting scheme $(Setup, BallotGen, VerifyBallot, VerifyAudit, Tally)$ is said to be auditable by voter if for all PPT adversary $\mathcal{B}$, $Adv_{\mathcal{B},SNDT}(\lambda)$ is negligible, where

$$Adv_{\mathcal{B},SNDT}(\lambda) = Pr[Exp_{\mathcal{B},SNDT}(\lambda) = 1]$$

$Exp_{\mathcal{B},SNDT}(\lambda)$
$(prm, n_{\max}, c, \mathcal{V}, \mathcal{T}, \mathfrak{T}, \mathcal{S}, \mathfrak{B}, st_{init}) \leftarrow Setup(\lambda, \alpha)$
$n \xleftarrow{\$} [1, n_{\max}]$
$st_1 \leftarrow st_{init}, i \leftarrow 1$
$Aux_1 = (n, c, \mathcal{V}, \mathcal{T}, \mathfrak{T}, \mathcal{S}, \mathfrak{B}, st_1)$
WHILE $(i \leq n)$
$(b_i, \tau_i, Aux_{i+1}) \leftarrow \mathcal{B}^{BallotGen, VerifyBallot, VerifyAudit, Tally}(prm, Aux_i)$
$v_i \leftarrow VerifyAudit(b_i, \tau_i)$
IF $(v_i = \perp)$
Abort
$i \leftarrow i + 1$
END WHILE
$st_{n+1} \leftarrow \mathcal{B}^{BallotGen, VerifyBallot, VerifyAudit, Tally}(prm, Aux_{n+1})$
$B \leftarrow (b_1, b_2, \ldots, b_n)$
$T \leftarrow Tally(prm, B, st_{n+1}, st_1)$
$(T \neq \perp) \wedge (T \neq f(v_1, v_2, \ldots, v_n))$

A DRE based e-voting scheme that is both universally verifiable and voter auditable is called an E2E verifiable e-voting scheme. The first property, i.e. universal verifiability assures that if a ballot passes the $VerifyBallot()$ test, there will be a trapdoor such that $VerifyAudit()$ function would return a valid vote, given the ballot and the trapdoor. That is, if all the ballots pass the $VerifyBallot()$ test, then all of them would correspond to a vote selected from the set $\mathcal{V}$. Hence, if all the ballots pass the verification test, then the tally will not include any inappropriate/out-of-range vote. The second property, i.e. auditability provides assurance that the ballots can be audited by the voters. Any voter may ask the DRE machine to reveal the trapdoor corresponding to a ballot, so she would be able to verify that the ballot produced against her vote indeed corresponds

to the correct choice she had made during voting. Every voter can check the correctness of her cast ballot by asking to reveal the trapdoor. However, revealing the trapdoor makes the entire voting process susceptible to coercion attack. The voter can take the ballot and the corresponding trapdoor to a coercer to demonstrate that she indeed has voted in a particular way. In order to circumvent this, we adopt the method of voter initiated auditing by Benaloh [6]. We allow a voter to cast multiple ballots with the condition that the DRE will not reveal the trapdoor corresponding to her last ballot. However, she will receive the trapdoors corresponding to all other ballots. We call her last ballot a 'confirmed' ballot and all other previous ballots 'audited' ballots. There is no limit on how many ballots a particular voter can audit. The voter only chooses to audit or confirm her ballot once the ballot is printed on the receipt. If the voter chooses to audit her ballot, the DRE has to disclose the trapdoor that can reveal the vote encrypted in the ballot. Since, the DRE does not know beforehand whether the voter is going to confirm her ballot or audit it, the DRE cannot act maliciously and generate a wrong ballot that does not correspond to the choice the voter made previously without getting detected. When the voter confirms her ballot, the DRE does not reveal the corresponding trapdoor making coercion impossible. The DRE posts all audited and confirmed ballots on the Bulletin Board. The audited ballots are accompanied by their respective trapdoors. Hence, anyone can use the $VerifyAudit(\cdot)$ function to regenerate the votes corresponding to all the audited ballots. Then the audited ballots can be subtracted from the tally to obtain the tally constituted by the confirmed ballots only as stated in Section 4. Recall that the universal verifiability property assures that every ballot that passes the $VerifyBallot()$ test would correspond to a unique trapdoor that maps the ballot to a unique vote in $\mathcal{V}$. Thus, the two properties viz. universal verifiability and auditability together provide assurance that every valid ballot would contribute a unique vote to the final tally. A voter can use voter initiated auditing method to confirm with a high degree of confidence that this vote is same as the one she entered to the DRE machine.

The following lemma proves that no PPT adversary can generate a final state such that the tally function returns a different tally that does not conform to the votes that the ballots correspond to. This means that once the voting is over, the adversary cannot generate a final state st so that the tally function returns an incorrect result(i.e. sum of votes).

Lemma 1. *Let us consider the following experiment* $Exp_{\mathcal{A},FST}(\lambda)$*. For any probabilistic polynomial time adversary* $\mathcal{A}$*,* $Pr\left[Exp_{\mathcal{A},FST}(\lambda) = 1\right] \leq negl(\lambda)$*.*

$Exp_{\mathcal{A},FST}(\lambda)$
$(n_{\max}, c, \mathcal{V}, \mathcal{T}, \mathfrak{T}, \mathcal{S}, \mathfrak{B}, st_{init}) \leftarrow Setup(\lambda, \alpha)$
$n \xleftarrow{\$} [1, n_{\max}],\ st_1 \leftarrow st_{init}, i \leftarrow 1$
$V = (v_1, v_2, \ldots, v_n) \xleftarrow{\$} \mathcal{V}^n$
$Aux = (n, c, \mathcal{V}, \mathcal{T}, \mathfrak{T}, \mathcal{S}, \mathfrak{B}, st_1, V)$
$WHILE\ (i \leq n)$
$(b_i, \pi_i, \tau_i, st_{i+1}) \leftarrow BallotGen(prm, v_i, st_i)$
$i \leftarrow i + 1$
$END\ WHILE$
$B \leftarrow (b_1, b_2, \ldots, b_n)$
$\Pi \leftarrow (\pi_1, \pi_2, \ldots, \pi_n)$
$\tau \leftarrow (\tau_1, \tau_2, \ldots, \tau_n)$
$\mathbb{S} \leftarrow (st_2, st_3, \ldots, st_{n+1})$
$st \leftarrow \mathcal{A}^{BallotGen, VerifyBallot, VerifyAudit, Tally}(B, \Pi, \Gamma, \tau, \mathbb{S}, Aux)$
$T = Tally(prm, B, st, st_1)$
$Return\ (T \neq \perp) \wedge (T \neq f(v_1, v_2, \ldots, v_n))$

Proof. This property follows directly from the auditability property. If there exists an adversary $\mathcal{A}$, such that $Pr[Exp_{\mathcal{A},FST}(\lambda) = 1] > negl(\lambda)$, then we can construct another adversary $\mathcal{B}$ against the security experiment $Exp_{\mathcal{B},SNDT}(\lambda)$. $\mathcal{B}$ can generate all the ballots by simulating the $BallotGen()$ algorithm with arbitrary votes. The algorithm returns the encrypted ballot, proof of well-formedness, the trapdoor , and the updated states whenever it is invoked with some input. $\mathcal{B}$ returns the ballot, the trapdoor and the updated state. When n number of votes have been cast, the adversary $\mathcal{B}$ invokes $\mathcal{A}$ with the required inputs. $\mathcal{A}$ returns a st such that $T = Tally(prm, B, st, st_1)$ holds. In addition $T \neq \perp$, and $T \neq f(v_1, v_2, \ldots, v_n)$ also hold. Now, $\mathcal{B}$ can return the same st. It is easy to see that $\mathcal{B}$ will win the experiment $Exp_{\mathcal{B},SNDT}(\lambda)$ if $\mathcal{A}$ wins the $Exp_{\mathcal{A},FST}(\lambda)$ experiment. Hence, the result holds.

3.3 Security against Malicious Bulletin Board

Each ballot posted by the DRE machine onto the Bulletin Board is digitally signed by the DRE machine itself. This prevents an adversary that controls the Bulletin Board from inserting new items on the Bulletin Board or altering existing ones. However, the Bulletin Board may attempt to drop one or more ballots. A robust e-voting scheme should ensure that the Bulletin Board cannot drop ballots without detection. Consider the following security experiment $Exp_{\mathcal{A},MBB}(\lambda)$. The advantage of the adversary $\mathcal{A}$, against the security experiment $Exp_{\mathcal{A},MBB}(\lambda)$ is written as $Adv_{\mathcal{A},FST}(\lambda) = Pr[Exp_{\mathcal{A},MBB}(\lambda) = 1]$. A DRE based e-voting scheme $(Setup, BallotGen, VerifyBallot, VerifyAudit, Tally)$ is said to be secure against malicious Bulletin Board if for all PPT adversary $\mathcal{B}$, $Adv_{\mathcal{B},MBB}(\lambda)$ is negligible. This property ensures that if the Bulletin Board drops some ballots, it would not go undetected as in such cases, the tally function would return error symbols. Note that this property may not be required

for an e-voting system if the Bulletin Board is considered to be secure against adversarial attack.

$Exp_{\mathcal{A},MBB}(\lambda)$
$(n_{\max}, c, \mathcal{V}, \mathcal{T}, \mathfrak{T}, \mathcal{S}, \mathfrak{B}, st_{init}) \leftarrow Setup(\lambda, \alpha)$
$n \xleftarrow{\$} [1, n_{\max}]$, $st_1 \leftarrow st_{init}, i \leftarrow 1$
$V = (v_1, v_2, \ldots, v_n) \xleftarrow{\$} \mathcal{V}^n$
$Aux = (n, c, \mathcal{V}, \mathcal{T}, \mathfrak{T}, \mathcal{S}, \mathfrak{B}, st_1, V)$
WHILE $(i \leq n)$
$(b_i, \pi_i, \tau_i, st_{i+1}) \leftarrow BallotGen(prm, v_i, st_i)$
$i \leftarrow i + 1$
END WHILE
$B \leftarrow (b_1, b_2, \ldots, b_n)$
$\Pi \leftarrow (\pi_1, \pi_2, \ldots, \pi_n)$
$\tau \leftarrow (\tau_1, \tau_2, \ldots, \tau_n)$
$B' \leftarrow \mathcal{A}^{BallotGen, VerifyBallot, VerifyAudit, Tally}(B, \Pi, \Gamma, \tau, st_{n+1}, Aux)$
$T = Tally(prm, B', st, st_{n+1})$
Return $(T \neq \perp) \wedge (B' \subset B)$

3.4 Voter Privacy

Here, we shall define the privacy property of a DRE based e-voting system. We shall be using almost the same definition of privacy which was proposed by Benaloh in [9]. However, the definition provided in [9] is aimed for e-voting systems that require tallying authorities, hence, we have made a minor modification to the original definition due to the fact that our e-voting technique does not require any tallying authority. The definition of Benaloh requires a key pair of the challenger that emulates the real-world tallying authority. The votes are encrypted using the public key. The adversary uses the voting oracle to cast ballots. In each attempt, the adversary provides a pair of votes: v_0, and v_1. The challenger chooses either v_0 or v_1 depending upon a pre-selected bit d. Once the adversary has cast enough ballots, the challenger checks whether the tally would be the same for $d = 0$, and $d = 1$ or not. If they are not the same, the challenger aborts and the adversary fails. If they are, the challenger generates the tally and a NIZK proof of correctness of the tally. Now, the adversary has to guess the value of d. Here, we shall keep the fundamental precepts of Benaloh's definition intact. However, it should be noted that unlike Benolah's definition of voter privacy, our scheme does not have any key pair of tallying authorities; rather in our scheme the DRE machine generates the ballots depending on its current state, and a chosen randomness. As a result of the generation of a ballot, the DRE machine needs to update its state. Once all the ballots are generated, the DRE machine publishes its final state. This is precisely the difference between our definitions of voter privacy and that of Benaloh's.

In our security experiment there is a challenger and an adversary. We consider a PPT adversary $\mathcal{A}$ who can compromise an arbitrary number of voters. $\mathcal{A}$ can

influence the compromised voters to vote the way she wants. She can choose their votes, the number of audits she wants each of them to make etc. The challenger emulates the uncompromised voters. We also give $\mathcal{A}$ the freedom to choose two sets of votes (e.g. V_0 and V_1) for the uncorrupted voters. As such our security notion is stronger than what is required of an e-voting scheme as in a normal election the adversary does not get to choose the votes of uncompromised voters. These votes are chosen by honest voters (or by the challenger) themselves. So if any e-voting scheme is secure under this security notion, it will definitely be secure in a real world scenario where the adversary has no control over the selection of votes by the honest voters. We require that the tally constituted by V_0 and V_1 as votes of honest voters (chosen by $\mathcal{A}$) should be equal, as otherwise distinguishing between them would be trivial. The challenger flips a coin and depending on the outcome, selects a bit $b \in \{0, 1\}$. The challenger uses V_b as the set of votes for the uncompromised voters. The corrupt voters and the honest voters may come in any arbitrary order chosen by the adversary $\mathcal{A}$. Once, the protocol finishes, all the ballots are posted on the public Bulletin Board. Now, the adversary has to guess the bit b. If she can guess b correctly, she wins the game. We call the game $Exp_{\mathcal{A},IND-CVA}(\lambda)$. The advantage of $\mathcal{A}$ against this security game is $Adv_{\mathcal{A},IND-CVA}(\lambda)$. Let us assume that $Exp_{\mathcal{A},IND-CVA}(\lambda)$ equals 1 only when adversary can guess b correctly. As such,

$$Adv_{\mathcal{A},IND-CVA}(\lambda) = Pr[Exp_{\mathcal{A},IND-CVA}(\lambda) = 1] - \frac{1}{2}$$

Remark: In the experiment $Exp_{\mathcal{A},IND-CVA}(\lambda)$, the adversary gets to choose two distinct sets of votes for all the voters, with the condition that the tally for these two sets should be the same. These two sets are V_0 and V_1. Note that the intersection of these two sets may not be empty. That is some votes in V_0 and V_1 may be identical. However all the votes must not be the same, otherwise the adversary would not need to distinguish between them. The e-voting scheme is secure against chosen vote attack if for all PPT adversary $\mathcal{A}$,

$$Adv_{\mathcal{A},IND-CVA}(\lambda) \leq negl(\lambda)$$

4 Voting Procedure

In this section, we describe a practical voting process. In DRE based voting schemes, every voter has to go to a polling kiosk to cast her vote. Each voter brings identification documents for authentication and after successful authentication obtains a random credential (a passcode or a smart-card) which allows them to log onto a DRE to cast one vote. The DRE is programmed with the total number of allowable votes N (including audited ballots), the actual number of voters n ($n < N$), the number of candidates, the set of acceptable votes $\mathcal{V}$, the set of trapdoors $\mathfrak{T}$, the state space $\mathcal{S}$, the ballot space $\mathfrak{B}$, and the initial state $st_1 = st_{init}$. Apart from these, the DRE also has access to the function $BallotGen(\cdot)$. Initially st_1 is posted on the Bulletin Board. In an iterative stage

$i \in [1, N]$, let us assume that the d'th voter ($d \in [1, n]$) is casting her ballot. The DRE machine provides an interactive interface to the voter so she could specify her voting choice. Once the voter has input her vote v_i, the DRE machine invokes $BallotGen(v_i, st_i)$. Here st_i is the current state of the DRE. The function $BallotGen(\cdot)$ returns $(b_i, \pi_i, \tau_i, st_{i+1})$. The DRE prints b_i and π_i on the paper. The BallotGen function should include the ballot id i in the construction of π_i in order to defend against clash attack. Then the DRE provides two options to the voter: either audit her ballot or to confirm it. If the voter chooses to audit her ballot, the DRE prints τ_i on paper. The DRE makes an entry on the Bulletin Board of the form $(\text{Audited}, i, b_i, \tau_i)$. Then the DRE updates her state to st_{i+1}. The DRE then starts over again, and lets the voter make a fresh choice again. The voter may choose to audit as many times as she wants provided the total number ballots generated is less than $n_{\max}$. In the end, the voter makes a final choice and proceeds to confirm her vote. If the voter confirms her vote v_i, the DRE machine posts $(\text{Confirmed}, i, b_i, \pi_i)$ on the Bulletin Board. The voter receives the printed receipt containing all the items printed by the DRE machine since she started interacting with it. This includes data corresponding to her audited ballots and the data corresponding to her final confirmed ballot. Once, all the voters have cast their votes, the DRE machine posts its final state st_N on the Bulletin Board. Here, we assume that the total number of audited and confirmed ballots on the Bulletin Board is N. Anyone can compute $T = Tally\,(B, st_N, st_1)$, where B is the set of all ballots. Let D be the set of indices of all audited ballots posted on the Bulletin Board. Anyone can also compute $\tilde{v}_j = VerifyAudit(b_j, \tau_j)$ for all $j \in D$. The tally is v, where $v = T - \sum_{j \in D} \tilde{v}_j$.

Lemma 2. *Let there be n voters in a certain election. The voters are designated as $V_i : i \in [1, n]$. Each voter V_i makes n_i audits, before casting her final ballot. Let, P be probability that the DRE generates a ballot fraudulently during casting of a ballot. As such, the probability that the DRE machine will successfully be able to cast at least one inexact confirmed ballot leading to an incorrect tally is given by $(1-P)^A\,(1-(1-P)^n)$, where A is the total number of audits done by all the voters.*

Proof. The DRE will be caught if it generates a wrong ballot which the voter chooses to audit. Let, X_i denote the event that the i'th ballot is fraudulently generated by the DRE machine. Since, each voter V_i makes n_i audits before confirming a ballot, the probability we are intersted in is as follows:

$$K = Pr\left[\begin{array}{c} X_1^c \cap X_2^c \cap \ldots \cap X_{n_1}^c \cap \\ X_{n_1+2}^c \cap X_{n_1+3}^c \cap \ldots \cap X_{n_1+n_2+1}^c \cap \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ X_{\sum_{i=1}^{n-1} n_i+n}^c \cap X_{\sum_{i=1}^{n-1} n_i+n+1}^c \cap \ldots \cap X_{\sum_{i=1}^{n} n_i+n-1}^c \cap \\ \left(X_{n_1+1}^c \cap X_{n_1+n_2+1}^c \cap \ldots X_{\sum_{i=1}^{n} n_i+1}^c\right)^c \end{array}\right]$$

Since, X_i's are independent events, and $Pr[X_i] = P$ for all $i \in [1, \sum_{i=1}^{n} n_i + n]$, we have

$$K = (1-P)^{\sum_{i=1}^{n} n_i} \left(1 - (1-P)^n\right)$$

Since, $\sum_{i=1}^{n} n_i$ is the total number of audits done by all the n voters,

$$K = (1-P)^A \left(1 - (1-P)^n\right),$$

where A is the total number of audits.

Differentiating with respect to P, we get

$$\frac{\partial K}{\partial P} = n(1-P)^A (1-P)^{n-1} - A\left(1 - (1-P)^n\right)(1-P)^{A-1}$$

If K is maximum at $P = P_{\max}$, then $\frac{\partial K}{\partial P}\big|_{P=P_{\max}} = 0$. Solving the above equation, we get, $(1 - P_{\max})^n = \frac{A}{A+n}$. Let us assume that the total number of audits $A = \kappa n$. Putting $A = \kappa n$, we get $(1-P_{\max})^n = \frac{\kappa}{\kappa+1}$. Thus, $P_{\max} = 1-(\frac{\kappa}{\kappa+1})^{1/n}$. Hence, $K_{\max} = (1-P_{\max})^{\kappa n}\left(1 - (1-P_{\max})^n\right) = \frac{\kappa^\kappa}{(\kappa+1)^{(\kappa+1)}}$. Figure 1 shows how $K_{\max}$ changes with the change of the value of κ.

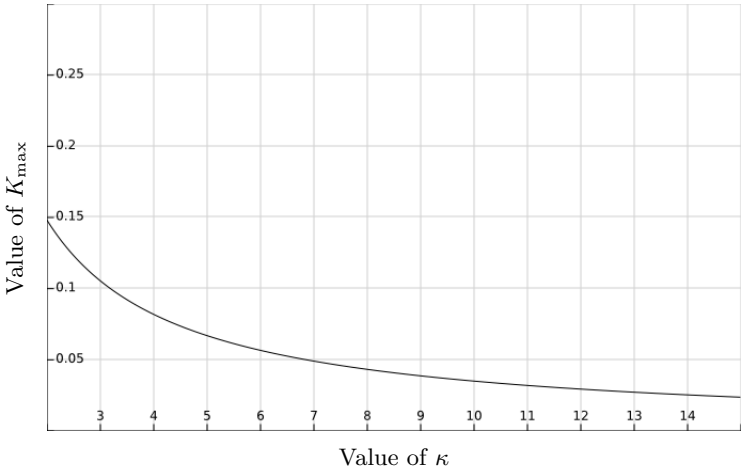

Fig. 1. Graphical presentation of the variation in the value of $K_{\max}$ with respect to the variation in κ.

5 Discussion

Here we provide a detailed discussion on the properties of our scheme. We consider verifiability by an individual voter, and global verifiability. We also consider attacks on the Bulletin Board and coercion resistance. We also discuss the robustness of this scheme.

Verification by Voter Our scheme provides E2E verifiability of each of the cast vote. Hence, each voter would be able to verify that her vote was cast and included in the tally as per her wish. Further, any third party would be able to check that none of the ballots that constitute the tally corresponds to out-of-range vote. The former goal is achieved thanks to the auditability property. This property ensures that the tally must be constituted of the votes obtained through decryption of the votes with the help of their corresponding trapdoors. Again the uniqueness property of trapdoors ensure that their can be only one trapdoor corresponding to one ballot with which it could be decrypted. So, a voter can do sufficient number of audits to ensure that her vote is correctly encapsulated in her final confirmed ballot. The DRE machine does not know at the time of printing the receipt whether the voter is going to audit her ballot or not, so she cannot cheat on the voter without risking getting caught. Thus, when the polling ends, the voter can assure herself that her vote is properly cast, and counted into the tally by performing sufficient number of audits and by matching her final receipt with the one posted on the Bulletin Board.

Public Verifiability A vote is the secret information of a voter. No one else should get to learn it. However, for the sake of verifiability, third parties should be able to at least verify that all the cast ballots correspond to votes that are within the permissible limit, and there is no ballot that correspond to out-of-range vote. This is done by using the non-interactive proofs π that demonstrate the well-formedness of each ballot. Given a ballot and a proof π, the $VerifyBallot$ function returns true with overwhelming probability only if the vote that corresponds to the ballot belongs to the set $\mathcal{V}$. Hence, the tally computed from the ballots are constituted by valid votes. The DRE should include the id of each ballot in the construction of the proof π in order to defend against clash attack [31].

Bulletin Board The Bulletin Board, used as a storage, is secure against Denial of Service attack. Note that, insertion of any new entry by the adversary would require the adversary to know the signature-key of the DRE machine. So, the adversary cannot alter existing data on the Bulletin Board if the signature is not malleable. If the adversary drops one or more ballots from the Bulletin Board, the verification would fail which will alert the authorities. This is due to the fact that for any PPT adversary $Adv_{\mathcal{A},MBB}(\lambda) \leq negl(\lambda)$.

Robustness In our scheme, all the ballots and other information are generated by the DRE machine. The voter only uses the DRE machine's interface to enter her choice. Moreover, the voter does not have the secret signing key of the DRE machine. Hence, she cannot generate arbitrary receipts. Therefore, a malicious voter cannot halt the election process anyway.

6 DRE-CV

Here, we discuss our new DRE based cumulative e-voting system. We have used the DRE-ip scheme by Shahandashti and Hao [39] as a building block. The DRE-

ip scheme is designed for simple plurality voting. There, the votes have two simple values: 0 or 1. In order to adapt it for supporting cumulative voting which is a type of ranked-choice electoral voting systems, we need to construct different non-interactive zero knowledge proof in order to make it universally verifiable. Below, we formally define the construction according to our formalization.

The scheme consists of five algorithms who work as below:

□ *Setup*(λ, α) This algorithm takes the security parameter λ, and the total number of points α available to each of the voters. The algorithm returns the maximum number of votes $n_{\max}$, the number of candidates c, the set of valid votes

$$\mathcal{V} = \left\{ (v_1, v_2, \ldots, v_c) : v_1, v_2, \ldots, v_c \in [\alpha], \sum_{i=1}^{c} v_i = \alpha \right\}$$

the Tally space

$$\mathcal{T} = \left\{ \sum_{i=1}^{n} v_i : v_i \in \mathcal{V}, n \in [1, n_{\max}] \right\}$$

the set of trapdoors

$$\tau = \{x_1, x_2, \ldots, x_c : x_i \in \mathbb{Z}_p\}$$

the set of states

$$\mathcal{S} = \tau$$

the set of ballots

$$\mathfrak{B} = \{G^c \times G^c\}$$

and the initial state $st_{init} = (0, 0, \ldots, 0)$. It assigns $prm = (G, p, g, \tilde{g})$, where g and $\tilde{g}$ are two generators of G, such that the relationship between them is not known.

□ *BallotGen*(v, st_{in}) This algorithm takes a vote $v \in \mathcal{V}$ and a state st_{in}. It parses the vote v as $(v_1, v_2, \ldots, v_c)$ and the state st_{in} as $(st_1, st_2, \ldots, st_c)$. The algorithm selects random $X = (x_1, x_2, \ldots, x_c) \in \mathbb{Z}_p^c$. Then it assigns $b_i = g^{x_i + v_i}$, and $\tilde{x}_i = \tilde{g}^{x_i}$. It generates a NIZK proof $\mathit{\Pi} = (\pi_1, \pi_2, \ldots, \pi_c)$ and $\tilde{\mathit{\Pi}}$ of well-formedness of the ballot. Each proof $\pi_i, i \in [1, c]$ proves that $b_i = DH_{\tilde{g}}(g, \tilde{x}_i) * g^{K_i}$ for some $K_i \in [\alpha]$. In addition $\tilde{\mathit{\Pi}}$ proves that

$$\prod_{i=1}^{c} b_i = DH_{\tilde{g}}\left(g, \prod_{i=1}^{c} \tilde{x}_i\right) * g^{\alpha}$$

Thus, together $\mathit{\Pi}$ and $\tilde{\mathit{\Pi}}$ proves that there exists $v = (v_1, v_2, \ldots, v_c) \in \mathcal{V}$, such that $b_i = g^{x_i + v_i}$, and $\tilde{x}_i = \tilde{g}^{x_i}$. Details about the NIZK proof construction can be found in Appendix. Let $B = (b_1, b_2, \ldots, b_c)$, and $\tilde{X} = (\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_c)$. Let $\tilde{b} = \langle B, \tilde{X} \rangle$, $\bar{\pi} = (\mathit{\Pi}, \tilde{\mathit{\Pi}})$, $\tau = X$, and $st_{out} = (st'_1, st'_2, \ldots, st'_c)$, where $st'_k = st_k + x_k$ for all $k \in [1, c]$. The algorithm returns $(\tilde{b}, \bar{\pi}, \tau, st_{out})$.

- $VerifyAudit(\tilde{b}, \tau)$ This algorithm takes as input a ballot $\tilde{b}$, and a trapdoor τ. It parses $\tilde{b}$ as $\langle B, \tilde{X} \rangle$, and τ as $(x_1, x_2, \ldots, x_c)$. Here, $B = (b_1, b_2, \ldots, b_c)$, and $\tilde{X} = (\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_c)$. If the parsing cannot be done properly, it returns $\perp$. Else it computes $v = (v_1, v_2, \ldots, v_c) \in \mathbb{Z}_p^c$, such that $b_i = g^{x_i + v_i}$, and $\tilde{x}_i = \tilde{g}^{x_i}$, for all $i \in [1, c]$. Then it returns v. Else it returns $\perp$.
- $VerifyBallot(\tilde{b}, \bar{\pi}, \mathcal{V})$ This algorithm takes as input a ballot $\tilde{b}$, a proof of well-formedness $\bar{\pi}$. It first parses $\tilde{b}$ as $\langle B, \tilde{X} \rangle$, and $\bar{\pi}$ as $(\Pi, \tilde{\Pi})$. Here, $B = (b_1, b_2, \ldots, b_c)$, $\tilde{X} = (\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_c)$. If the parsing cannot be done properly, it returns $False$. Then Π is parsed as $(\pi_1, \pi_2, \ldots, \pi_c)$. If the parsing cannot be done, it returns $False$. Else it checks whether or not π_i is a valid NIZK proof for the following statement:

  $$\exists K_i \in [\alpha] : b_i = DH_{\tilde{g}}(g, \tilde{x}_i) * g^{K_i}$$

  for all $i \in [1, c]$. If the check is not successful for at least one value of $i \in [1, c]$, the algorithm returns $False$. Else the algorithm checks whether $\tilde{\Pi}$ is a valid NIZK proof for the following statement:

  $$\prod_{i=1}^{c} b_i = DH_{\tilde{g}}\left(g, \prod_{i=1}^{c} \tilde{x}_i\right) * g^{\alpha}$$

  If the check is successful, the algorithm return $True$, otherwise it returns $False$.
- $Tally(\mathbf{B}, st_f, st_1)$ This function takes as input the set of ballots $\mathbf{B} = (\tilde{b}_1, \tilde{b}_2, \ldots, \tilde{b}_c)$, the initial state $st_1 = (0, 0, \ldots, 0)$, and the final state st_f. It parses st_1 as $(s_1, s_2, \ldots, s_c)$. If the parsing cannot be done, it returns $\perp$. Else it parses $\tilde{b}_i$ as $\langle B_i, \tilde{X}_i \rangle$, for $i \in [1, n]$. Here, $B_i = (b_{i1}, b_{i2}, \ldots, b_{ic})$, and $\tilde{X}_i = (\tilde{x}_{i1}, \tilde{x}_{i2}, \ldots, \tilde{x}_{ic})$. The algorithm checks whether

  $$\prod_{i=1}^{n} \tilde{x}_{ij} = \tilde{g}^{s_j}; \forall j \in [1, c]$$

  If the check is not successful it returns $\perp$. Else, it find $(v_1, v_2, \ldots, v_c) \in \mathcal{T}$, such that

  $$\prod_{i=1}^{n} b_{ij} = g^{s_j + v_j}$$

  This check can be done in polynomial time as $|\mathcal{T}| \in poly(\lambda)$. If it can find such $(v_1, v_2, \ldots, v_c) \in \mathcal{T}$, it returns the same. If the above check is unsuccessful, it returns $\perp$.

6.1 Correctness

Now, we show that the above cumulative e-voting scheme is correct. As discussed before, the scheme will be correct if the following properties are satisfied:

1. If $BallotGen(prm, v_i, st)$ algorithms returns $(\tilde{b}_i, \pi_i, \tau_i, st')$, then $VerifyBallot(prm, \tilde{b}_i, \pi_i, \mathcal{V})$ should return $True$, and $VerifyAudit(prm, \tilde{b}_i, \tau_i)$ should return v_i. In our proposed scheme, $\tilde{b}_i = \langle B_i, \tilde{X}_i \rangle$, $B_i = (b_{i1}, b_{i2}, \ldots, b_{ic})$, and $\tilde{X}_i = (\tilde{x}_{i1}, \tilde{x}_{i2}, \ldots, \tilde{x}_{ic})$. Again, $b_{ij} = DH_{\tilde{g}}(g, \tilde{x}_{ij}) * g^{v_{ij}}$, and $v_i = (v_{i1}, v_{i2}, \ldots, v_{ic})$, for $j \in [1, c]$. In our scheme, π_i is the non-interactive zero knowledge proof of the fact that $(K_1, K_2, \ldots, K_c) \in \mathcal{V}$, where $g^{K_j} = b_{ij}/DH_{\tilde{g}}(g, \tilde{x}_{ij})$ for $j = 1, 2, \ldots, c$. Hence, $VerifyBallot(prm, \tilde{b}_i, \pi_i, \mathcal{V}) = True$. Moreover, $\tau_i = (x_{i1}, x_{i2}, \ldots, x_{ic})$, where $x_{ij} = \log_{\tilde{g}} \tilde{x}_{ij}$, for $j \in [1, c]$. As such, $VerifyAudit(prm, \tilde{b}_i, \tau_i) = v_i = (v_{i1}, v_{i2}, \ldots, v_{ic})$.
2. Lastly, if $(\tilde{b}_i, \pi_i, \tau_i, st_{i+1}) \leftarrow BallotGen(prm, v_i, st_i)$, for $i = 1, 2, \ldots, n$, then $Tally(prm, \mathbf{B}, st_{i+1}, st_1)$ must return the tally $f(v_1, v_2, \ldots, v_n)$, where $\mathbf{B} = \{\tilde{b}_i : i \in [1, n]\}$. Now, in our scheme, $\tilde{b}_i = \langle B_i, \tilde{X}_i \rangle$, $B_i = (b_{i1}, b_{i2}, \ldots, b_{ic})$, $\tilde{X}_i = (\tilde{x}_{i1}, \tilde{x}_{i2}, \ldots, \tilde{x}_{ic})$, $b_{ij} = DH_{\tilde{g}}(g, \tilde{x}_{ij}) * g^{v_{ij}}$. Again, $st_1 = (0, 0, \ldots, 0)$, and $st_{n+1} = (s_1, s_2, \ldots, s_c)$, where $s_j = \sum_{i=1}^{n} x_{ij}$, and $x_{ij} = \log_{\tilde{g}} \tilde{x}_{ij}$: for all $j \in [1, c]$. As discussed before, the tally function returns $L = (l_1, l_2, \ldots, l_c)$, where $l_j = \log_g (\prod_{i=1}^{n} b_{ij}/g^{s_j}) = \log_g (\prod_{i=1}^{n} b_{ij}/DH_{\tilde{g}}(g, \tilde{g}^{s_j})) = \log_g (\prod_{i=1}^{n} b_{ij}/DH_{\tilde{g}}(g, \prod_{i=1}^{n} \tilde{x}_{ij})) = \log_g (\prod_{i=1}^{n} (DH_{\tilde{g}}(g, \tilde{x}_{ij}) * g^{v_{ij}}) / \prod_{i=1}^{n} DH_{\tilde{g}}(g, \tilde{x}_{ij})) = \sum_{i=1}^{n} v_{ij}$.Thus, the tally $L = (l_1, l_2, \ldots, l_c) = (\sum_{i=1}^{n} v_{i1}, \sum_{i=1}^{n} v_{i2}, \ldots, \sum_{i=1}^{n} v_{ic}) = f(v_1, v_2, \ldots, v_c)$.

Hence, the e-voting scheme is correct.

6.2 Uniqueness of the Trapdoor

We have discussed binding property of trapdoor previously in section 3. The above cumulative e-voting protocol will follow this property if for all PPT adversary $\mathcal{A}$, the following probability is negligible

$$Pr[Exp_{\mathcal{A},TBND}(\lambda) = 1]$$

$Exp_{\mathcal{A},TBND}(\lambda)$
$(prm, n_{\max}, c, \mathcal{V}, \mathcal{T}, \mathfrak{T}, \mathcal{S}, \mathfrak{B}, st_{init}) \leftarrow Setup(\lambda, \alpha)$
$Aux = (n_{\max}, c, \mathcal{V}, \mathcal{T}, \mathfrak{T}, \mathcal{S}, \mathfrak{B}, st_{init})$
$(\tilde{b}, \tau, \tau') \leftarrow \mathcal{A}^{BallotGen, VerifyBallot, VerifyAudit, Tally}(prm, Aux)$
$v = VerifyAudit(\tilde{b}, \tau)$
$v' = VerifyAudit(\tilde{b}, \tau')$
Return $(v \neq \perp) \wedge (v' \neq \perp) \wedge (v \neq v')$

Let us assume $v = (v_1, v_2, \ldots, v_c)$, and $v' = (v'_1, v'_2, \ldots, v'_c)$. If $Pr[Exp_{\mathcal{A},TBND}(\lambda) = 1] \not\leq negl(\lambda)$, there exist $\tilde{b} = (B, \tilde{X}), B = (b_1, b_2, \ldots, b_c), X = (\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_c)$ $\tau = (x_1, x_2, \ldots, x_c), \tau' = (x'_1, x'_2, \ldots, x'_c)$, satisfying

$$b_i = DH_{\tilde{g}}(g, \tilde{x}_i) * g^{v_i}, \forall i \in [1, c]$$

$$b_i = DH_{\tilde{g}}(g, \tilde{x}_i) * g^{v'_i}, \forall i \in [1, c]$$

and

$$\tilde{x}_i = \tilde{g}^{x_i} = \tilde{g}^{x'_i}, \forall i \in [1, c]$$

This undoubtedly means that $x_i = x'_i$, and $v_i = v'_i, \forall i \in [1, c]$. Hence, $v = v'$. Thus, our scheme exhibits the uniqueness property of trapdoors.

6.3 Universal Verifiability

The above scheme will exhibit universal verifiability if the below probability is negligible.

$$Adv_{\mathcal{A},SNDN}(\lambda) = Pr[Exp_{\mathcal{A},SNDN}(\lambda) = 1]$$

$Exp_{\mathcal{A},SNDN}(\lambda)$
$(prm, n_{\max}, c, \mathcal{V}, \mathcal{T}, \mathfrak{T}, \mathcal{S}, \mathfrak{B}, st_{init}) \leftarrow Setup(\lambda, \alpha)$
$Aux = (n_{\max}, c, \mathcal{V}, \mathcal{T}, \mathfrak{T}, \mathcal{S}, \mathfrak{B}, st_{init})$
$(\tilde{b}, \bar{\pi}) \leftarrow \mathcal{A}^{BallotGen, VerifyBallot, VerifyAudit, Tally}(prm, Aux)$
$(VerifyBallot(\tilde{b}, \bar{\pi}, \mathcal{V}) = True) \wedge (\forall \tau \in \mathfrak{T} : VerifyAudit(b, \tau) = \perp)$

If the NIZK proof system is sound and if $VerifyBallot(\tilde{b}, \tilde{\pi}, \mathcal{V}) = True$, where $\tilde{b} = \langle B, \tilde{X} \rangle, B = (b_1, b_2, \ldots, b_c), \tilde{X} = (\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_c), \bar{\pi} = (\Pi, \tilde{\Pi})$, then for all $i \in [1, c]$, $\exists K_i \in [\alpha]$, such that $b_i = DH_{\tilde{g}}(g, \tilde{x}_i) * g^{K_i}$. In addition to that, $\prod_{i=1}^{c} b_i = DH_{\tilde{g}}(g, \prod_{i=1}^{c} \tilde{x}_i) * g^{\alpha}$. Let us assume that $x_i = \log_{\tilde{g}} \tilde{x}_i, \forall i \in [1, c]$. Also assume that $X = (x_1, x_2, \ldots, x_c)$. Again, $\prod_{i=1}^{c} b_i = DH_{\tilde{g}}(g, \prod_{i=1}^{c} \tilde{x}_i) * g^{\alpha} = \prod_{i=1}^{c} DH_{\tilde{g}}(g, \tilde{x}_i) * g^{\alpha}$. Hence, $\prod_{i=1}^{c} g^{K_i} = g^{\alpha}$, or $\sum_{i=1}^{c} K_i = \alpha$. Since $K_i \in [\alpha]$ for all $i \in [1, c]$, and $\sum_{i=1}^{c} K_i = \alpha$, $K = (K_1, K_2, \ldots, K_c) \in \mathcal{V}$. Thus, X is the trapdoor such that $VerifyAudit(\tilde{b}, X) = K \in \mathcal{V}$. Hence, our scheme is universally verifiable.

6.4 Auditability

Here, we show that our scheme is auditable by the voter.

Our scheme $(Setup, BallotGen, VerifyBallot, VerifyAudit, Tally)$ will be auditable by voter if for all PPT adversary $\mathcal{A}$, $Adv_{\mathcal{A},SNDT}(\lambda)$ is negligible, where

$$Adv_{\mathcal{A},SNDT}(\lambda) = Pr[Exp_{\mathcal{A},SNDT}(\lambda) = 1]$$

$Exp_{\mathcal{B},SNDT}(\lambda)$
$(prm, n_{\max}, c, \mathcal{V}, \mathcal{T}, \mathfrak{T}, \mathcal{S}, \mathfrak{B}, st_{init}) \leftarrow Setup(\lambda, \alpha)$
$n \xleftarrow{\$} [1, n_{\max}]$
$st_1 \leftarrow st_{init}, i \leftarrow 1$
$Aux_1 = (n, c, \mathcal{V}, \mathcal{T}, \mathfrak{T}, \mathcal{S}, \mathfrak{B}, st_1)$
WHILE $(i \leq n)$
$(\tilde{b}_i, \tau_i, Aux_{i+1}) \leftarrow \mathcal{A}^{BallotGen, VerifyBallot, VerifyAudit, Tally}(prm, Aux_i)$
$v_i \leftarrow VerifyAudit(\tilde{b}_i, \tau_i)$
IF $(v_i = \perp)$
Abort
$i \leftarrow i + 1$
END WHILE
$st_{n+1} \leftarrow \mathcal{A}^{BallotGen, VerifyBallot, VerifyAudit, Tally}(prm, Aux_{n+1})$
$\mathbf{B} \leftarrow (\tilde{b}_1, \tilde{b}_2, \ldots, \tilde{b}_n)$
$T \leftarrow Tally(prm, \mathbf{B}, st_{n+1}, st_1)$
$(T \neq \perp) \wedge (T \neq f(v_1, v_2, \ldots, v_n))$

Let us assume that for a PPT adversary $\mathcal{A}$, $Adv_{\mathcal{A},SNDT}(\lambda) \nleq negl(\lambda)$. Let us also assume that $st_{n+1} = (t_1, t_2, \ldots, t_n)$. Further, we assume $\tilde{b}_i = \langle B_i, \tilde{X}_i \rangle$, $B_i = (b_{i1}, b_{i2}, \ldots, b_{ic})$, and $\tilde{X}_i = (\tilde{x}_{i1}, \tilde{x}_{i2}, \ldots, \tilde{x}_{ic})$ for all $i = 1, 2, \ldots, n$. In addition, we assume that $v_i = (v_{i1}, v_{i2}, \ldots, v_{ic})$, and $\tau_i = (\tau_{i1}, \tau_{i2}, \ldots, \tau_{ic})$ for all $i \in [1, n]$. Since, $v_i = VerifyAudit(\tilde{b}_i, \tau_i), \forall i \in [1, n]$, we must have $\tilde{x}_{ij} = \tilde{g}^{x_{ij}}$, and $b_{ij} = g^{x_{ij}+v_{ij}}, \forall i \in [1, n], j \in [1, c]$. Let, $L = (l_1, l_2, \ldots, l_c)$ be the output of the tally function. Hence, $L \in \mathcal{T} \setminus \{f(v_1, v_2, \ldots, v_n)\}$ with a non-negligible probability. This can only happen if the following equations are satisfied:

$$\prod_{i=1}^{n} \tilde{x}_{ij} = \tilde{g}^{t_j}; \forall j \in [1, c]$$

and

$$\prod_{i=1}^{n} b_{ij} = g^{t_j + l_j}; \forall j \in [1, c]$$

Substituting $\tilde{x}_{ij}$ with $\tilde{g}^{x_{ij}}$, and b_{ij} with $g^{x_{ij}+v_j}$, we get $t_j = \sum_{i=1}^{n} x_{ij}$, and $\sum_{i=1}^{n}(x_{ij} + v_{ij}) = t_j + l_j$ for all $j \in [1, c]$. That is $\sum_{i=1}^{n} v_{ij} = l_j$, for all $j \in [1, c]$. Therefore, $Tally(prm, \mathbf{B}, st_{n+1}, st_1) = f(v_1, v_2, \ldots, v_n)$. Hence, our assumption was wrong, and the scheme is auditable by the voter.

6.5 Receipt-freeness

In our scheme, the DRE machine does not reveal the trapdoor corresponding to the final confirmed ballot of the voter. The receipt that a voter receives does not allow her to prove to someone how she voted. In fact, the receipt of a voter does not contain any sensitive information that is not available with the Bulletin Board. We shall prove that in section 7 that the Bulletin Board does not divulge any information about a voter's voting preference. Hence, our scheme is receipt-free.

6.6 Protection against Malicious Bulletin Board

We have discussed protection from malicious Bulletin Board in section 3. Our DRE-CV scheme will be secure against this attack if the tally function does not output a valid tally when some ballots are dropped. Let us assume that the set of ballots are $\{\tilde{b}_i : i \in [1,n]\}$, where $\tilde{b}_i = \langle B_i, \tilde{X}_i \rangle$, $B_i = (b_{i1}, b_{i2}, \ldots, b_{ic})$, and $\tilde{X}_i = (\tilde{x}_{i1}, \tilde{x}_{i2}, \ldots, \tilde{x}_{ic})$ for all $i \in [1,n]$. Hence, $\prod_{i=1}^{n} \tilde{x}_{ij} = \tilde{g}^{t_j}$, and $\prod_{i=1}^{n} b_{ij} = g^{t_j + v_j}$, for all $j \in [1,c]$. Here, the tally of n ballots is $(t_1, t_2, \ldots, t_n)$. Now, let us assume that K is the set of indices of the ballots which are dropped by the adversary, where $|K| \geq 1$. It is easy to see that $\prod_{i \in [1,n] \setminus K} \tilde{x}_{ij} \neq \tilde{g}^{t_j}$, for $j \in [1,c]$. Therefore, the tally function would return $\perp$, when at least one ballot is dropped. So, when a malicious adversary attempts to tamper with the bulletin board by dropping ballots or changing the tally, the attack will be publicly detectable.

7 Security Properties

Here, we discuss the security properties of the proposed scheme. We show that the proposed scheme is secure in the sense that it does not reveal any additional information regarding the votes of individual voters other than what the final tally normally does. For this purpose, we prove that our scheme is secure against the Chosen Vote Attack defined in section 3. In other words, no PPT adversary can make any inference vis-à-vis the secret ballots of honest voters which she cannot do from the tally or other information that she is entitled to have access to.

$Exp^{G}_{\mathcal{A},DDH}(\lambda)$
$g \xleftarrow{\$} G$
$A \xleftarrow{\$} G,\ B \xleftarrow{\$} G$
$c_0 = DH_g(A,B)$
$c_1 = g * DH_g(A,B)$
$b \xleftarrow{\$} \{0,1\}$
$b' = \mathcal{A}^{G}(g, A, B, c_b)$
return $(b = b')$

Assumption 1 *The advantage of the DDH adversary $\mathcal{A}$ is denoted as*

$$Adv^{G}_{\mathcal{A},DDH}(\lambda) = \left| Pr\left[Exp^{G}_{\mathcal{A},DDH}(\lambda) = 1\right] - \frac{1}{2} \right|$$

The above probability is taken over all random coins used in the experiment as well as those used by $\mathcal{A}$. If the Decisional Diffie Hellman assumption is intractable in G, then $Adv^{G}_{\mathcal{A},DDH}(\lambda) \leq negl(\lambda)$.

$Exp^{G}_{\mathcal{A},MDDH}(\lambda)$
$\kappa \xleftarrow{\$} poly(\lambda)$
$g \xleftarrow{\$} G$
$A \xleftarrow{\$} G$
$B_i \xleftarrow{\$} G : i \in [1,\kappa]$
$c_i^0 = DH_g(A, B_i) : i \in [1,\kappa]$
$c_i^1 = g * DH_g(A, B_i) : i \in [1,\kappa]$
$D_0 = (c_1^0, c_2^0, \ldots, c_\kappa^0)$
$D_1 = (c_1^1, c_2^1, \ldots, c_\kappa^1)$
$\mathbb{B} = (B_1, B_2, \ldots, B_\kappa)$
$b \xleftarrow{\$} \{0,1\}$
$b' = \mathcal{A}^G(g, A, \mathbb{B}, D_b)$
return $(b = b')$

Assumption 2 *The advantage of the MDDH adversary $\mathcal{A}$ is given by*

$$Adv^{G}_{\mathcal{A},MDDH}(\lambda) = \left| Pr\left[Exp^{G}_{\mathcal{A},MDDH}(\lambda) = 1\right] - \frac{1}{2} \right|$$

The above probability is taken over all random coins used in the experiment as well as those used by $\mathcal{A}$.

Lemma 3. $Adv^{G}_{\mathcal{B},MDDH}(\lambda) \leq Adv^{G}_{\mathcal{A},DDH}(\lambda)$.

Proof. Let, $\mathcal{B}$ be an adversary against the security experiment $Exp^{G}_{\mathcal{B},MDDH}(\lambda)$. We show how $\mathcal{A}$ can use it against $Exp^{G}_{\mathcal{A},DDH}(\lambda)$. $\mathcal{A}$ receives as inputs g, A, B and the challenge $c_b \in \{DH_g(A,B), g{*}DH_g(A,B)\}$. $\mathcal{A}$ selects random $d_1, d_2, \ldots, d_\kappa \in_R \mathbb{Z}_p$. $\mathcal{A}$ computes $B_i = B * g^{d_i}, c_i^b = c_b * A^{d_i}, \forall i \in [1,k]$. Thus, $\mathcal{A}$ sets $c_i^b = DH_g(A, B_i) * g^b$ for all $i \in [1,\kappa]$. Now, let us assume $D_b = (c_1^b, c_2^b, \ldots, c_\kappa^b)$ and $\mathbb{B} = (B_1, B_2, \ldots, B_\kappa)$. Note that, if $c_b = DH_g(A,B)$, then

$$D_b = (DH_g(A, B_1), DH_g(A, B_2), \ldots, DH_g(A, B_\kappa))$$

Alternatively, if $c_b = g{*}DH_g(A,B)$, then $D_b = (g{*}DH_g(A, B_1), g{*}DH_g(A, B_2), \ldots, g{*}DH_g(A, B_\kappa))$. Now, $\mathcal{A}$ invokes $\mathcal{B}$ with the following inputs: $g, A, \mathbb{B}$, and D_b. $\mathcal{B}$ will return a bit $b' \in \{0,1\}$. $\mathcal{A}$ will return the same bit b'. It is easy to see that the success probability of $\mathcal{A}$ is at least that of $\mathcal{B}$. Hence, the result holds.

Assumption 3 *Consider the following security experiment $Exp^{G}_{\mathcal{A},SDDH}(\lambda)$:*

Let us define the advantage of any PPT adversary $\mathcal{A}$ against this security experiment as

$$Adv^{G}_{\mathcal{A},SDDH}(\lambda) = \left| Pr\left[Exp^{G}_{\mathcal{A},SDDH}(\lambda) = 1\right] - \frac{1}{2} \right|$$

The above probability is taken over all random coins used in the experiment as well as those used by $\mathcal{A}$.

$Exp^{G}_{\mathcal{A},SDDH}(\lambda)$
$\eta, \alpha \xleftarrow{\$} poly(\lambda)$
$g \xleftarrow{\$} G, \tilde{g} \xleftarrow{\$} G$
$(V, V') = \mathcal{A}^{G}(\eta, \alpha, g, \tilde{g})$
if $((\lvert V\rvert \neq \eta) \lvert\lvert (\lvert V'\rvert \neq \eta) \lvert\lvert (\sum_{v \in V} v \neq \sum_{v' \in V'} v'))$
Abort
if $((\exists v \in V : v \notin [\alpha]) \lvert\lvert (\exists v' \in V' : v' \notin [\alpha]))$
Abort
$X_i \xleftarrow{\$} G : i \in [\eta]$
$c_i^0 = g^{v_i} * DH_{\tilde{g}}(g, X_i) : i \in [\eta]$
$c_i^1 = g^{v'_i} * DH_{\tilde{g}}(g, X_i) : i \in [\eta]$
$D_0 = (c_1^0, c_2^0, \ldots, c_\eta^0)$
$D_1 = (c_1^1, c_2^1, \ldots, c_\eta^1)$
$s = \sum_{i=1}^{\eta} DL_{\tilde{g}}(X_i)$
$\mathbb{X} = (X_1, X_2, \ldots, X_\eta)$
$b \xleftarrow{\$} \{0, 1\}$
$b' = \mathcal{A}^{G}(g, \tilde{g}, s, \mathbb{X}, D_b, V, V')$
return $(b = b')$

Lemma 4. $Adv^{G}_{\mathcal{B},SDDH}(\lambda) \leq Adv^{G}_{\mathcal{A},MDDH}(\lambda)$

Proof. Assume that $\mathcal{B}$ is an adversary against the security experiment $Exp^{G}_{\mathcal{B},SDDH}(\lambda)$. We show how the MDDH adversary $\mathcal{A}$ can use $\mathcal{B}$ against the MDDH assumption. $\mathcal{A}$ runs $\mathcal{B}$ internally as a subroutine. $\mathcal{A}$ receives as input $\tilde{g}, A = g, \mathbb{B} = (B_1, B_2, \ldots, B_{\eta-1})$, and a challenge $D_b = (c_1^b, c_2^b, \ldots, c_{\eta-1}^b)$, $c_i^b \in \{DH_{\tilde{g}}(g, \mathbb{B}), g * DH_{\tilde{g}}(g, \mathbb{B})\}$. Here, $DH_{\tilde{g}}(g, \mathbb{B}) = (DH_{\tilde{g}}(g, B_1), DH_{\tilde{g}}(g, B_2), \ldots, DH_{\tilde{g}}(g, B_{\eta-1}))$. $\mathcal{A}$ selects random $\alpha \xleftarrow{\$} poly(\lambda)$, and invokes $\mathcal{B}^{G}(\eta, \alpha, g, \tilde{g})$. $\mathcal{B}$ outputs $V = (v_1, v_2, \ldots, v_\eta)$ and $V' = (v'_1, v'_2, \ldots, v'_\eta)$, where $v_i, v'_i \in [\alpha]$ for all $i \in [1, \eta]$. Let us assume that $\sum_{i=1}^{\eta} v_i = \sum_{i=1}^{\eta} v'_i = \tilde{v}$. The adversary $\mathcal{A}$ computes $\tilde{c}_i^b = (c_i^b)^{v_i - v'_i} g^{v'_i}$ and $\tilde{X}_i = B_i^{v_i - v'_i}$ for all $i \in [1, \eta - 1]$. $\mathcal{A}$ also computes $\tilde{c}_\eta^b = \frac{g^{s+\tilde{v}}}{\prod_{i=1}^{\eta-1} c_i^b}$ and $\tilde{X}_\eta = \frac{\tilde{g}^s}{\prod_{i=1}^{\eta-1} X_i}$. Note that, if $c_i^b = DH_{\tilde{g}}(g, B_i)$ for all $i \in [\eta - 1]$, then $\tilde{c}_i^b = DH_{\tilde{g}}(g, X_i) * g^{v'_i}$ for all $i \in [\eta - 1]$. Alternatively, if $c_i^b = g * DH_{\tilde{g}}(g, B_i)$ for all $i \in [\eta - 1]$, then $\tilde{c}_i^b = DH_{\tilde{g}}(g, X_i) * g^{v_i}$ for all $i \in [\eta - 1]$. Similarly, if $c_i^b = DH_{\tilde{g}}(g, B_i)$, for all $i \in [\eta - 1]$, then $\tilde{c}_\eta^b = \frac{g^s}{DH_{\tilde{g}}(g, \prod_{i=1}^{\eta-1} X_i)} g^{v'_\eta} = \frac{g^s}{\prod_{i=1}^{\eta-1} DH_{\tilde{g}}(g, X_i)} g^{v'_\eta}$. Else if $c_i^b = g * DH_{\tilde{g}}(g, B_i), \forall i \in [1, \eta - 1]$, then $\tilde{c}_\eta^b = \frac{g^s}{DH_{\tilde{g}}(g, \prod_{i=1}^{\eta-1} X_i)} g^{v_\eta} = \frac{g^s}{\prod_{i=1}^{\eta-1} DH_{\tilde{g}}(g, X_i)} g^{v_\eta}$. Again, $s = DL_{\tilde{g}}(\prod_{i=1}^{\eta} X_i) = \sum_{i=1}^{\eta} DL_{\tilde{g}}(X_i)$. Let us assume $D_b = (\tilde{c}_1^b, \tilde{c}_2^b, \ldots, \tilde{c}_\eta^b)$ and $\mathbb{X} = (X_1, X_2, \ldots, X_\eta)$. Now, $\mathcal{A}$ invokes $\mathcal{B}$ with the following inputs: $g, \tilde{g}, s, \mathbb{X}, D_b, V, V'$. $\mathcal{B}$ will return a bit b'. $\mathcal{A}$ will return $1 - b'$.

It is easy to see that the success probability of $\mathcal{A}$ is at least that of $\mathcal{B}$, that is $Adv_{\mathcal{B}}^{G,SDDH}(\lambda) \leq Adv_{\mathcal{A}}^{G,MDDH}(\lambda)$.

Assumption 4 *Consider the following security experiment* $Exp_{\mathcal{A},LDDH}^{G}(\lambda)$*:*

$Exp_{\mathcal{A},LDDH}^{G}(\lambda)$
$n, \alpha \xleftarrow{\$} poly(\lambda)$
$g \xleftarrow{\$} G, \tilde{g} \xleftarrow{\$} G$
$(\eta, V_{n\times\eta}, V'_{n\times\eta}) = \mathcal{A}^{G}(n, \alpha, g, \tilde{g})$
if $(\eta < 2)$
Abort
$[v] = V_{n\times\eta}, [v'] = V'_{n\times\eta}$
if $(((\exists i \in [1,n]) \wedge (\exists j \in [1,\eta])) \wedge ((v_{ij} \notin [\alpha]) \vee (v'_{ij} \notin [\alpha])))$
Abort
if $((\exists j \in [1,\eta]) \wedge ((\sum_{i=1}^{n} v_{ij} \neq \alpha) \vee (\sum_{i=1}^{n} v'_{ij} \neq \alpha)))$
Abort
if $((\exists i \in [1,n]) \wedge (\sum_{j=1}^{\eta} v_{ij} \neq \sum_{j=1}^{\eta} v'_{ij}))$
Abort
$X \xleftarrow{\$} G^{\eta\times n}$
$X = [X_{ij}]$
$c_{ij}^{0} = g^{v_{ij}} * DH_{\tilde{g}}(g, X_{ij}) : j \in [1,\eta], i \in [1,n]$
$c_{ij}^{1} = g^{v'_{ij}} * DH_{\tilde{g}}(g, X_{ij}) : j \in [1,\eta], i \in [1,n]$
$D_0 = [c_{ij}^{0}]$
$D_1 = [c_{ij}^{1}]$
$s_i = \sum_{j=1}^{\eta} \log_{\tilde{g}}(X_{ij}) : i \in [1,n]$
$s = (s_1, s_2, \ldots, s_n)$
$b \xleftarrow{\$} \{0,1\}$
$b' = \mathcal{A}^{G}(g, \tilde{g}, s, X, D_b, V, V')$
return $(b = b')$

Let us define

$$Adv_{\mathcal{A}}^{G,LDDH}(\lambda) = \left| Pr\left[Exp_{\mathcal{A},LDDH}^{G}(\lambda) = 1\right] - \frac{1}{2} \right|$$

The above probability is taken over all random coins used in the experiment as well as those used by $\mathcal{A}$*.*

Lemma 5. $Adv_{\mathcal{B},LDDH}^{G}(\lambda) \leq poly(\lambda) * Adv_{\mathcal{A},SDDH}^{G}(\lambda)$.

Proof. We show that if there exists an adversary $\mathcal{B}$ against the security experiment $Exp_{\mathcal{B},LDDH}^{G}(\lambda)$, then it could be used to construct another adversary $\mathcal{A}$

against the security experiment $Exp^G_{\mathcal{B},SDDH}(\lambda)$. $\mathcal{A}$ works as below:

Upon receipt of $\eta, \alpha, g, \tilde{g}$, it selects random $n \xleftarrow{\$} poly(\lambda)$, and invokes $\mathcal{B}^G(n, \alpha, g, \tilde{g})$ and receives $\eta \in poly(\lambda)$, and two $n \times \eta'$ matrices $V = [v_{ij}]$ and $V' = [v'_{ij}]$. If $\eta' \neq \eta$, then $\mathcal{A}$ aborts and returns a random bit. Else, $\mathcal{A}$ continues. Let $\Delta = (\Delta_1, \Delta_2, \ldots, \Delta_n)$ be such that $\Delta_i = \sum_{j=1}^{\eta} v_{ij}$. $\mathcal{A}$ checks the dimension of both the matrices V and V'. If both the matrices are $n \times \eta$, then it continues, else it aborts and returns a random bit. If any of the elements in either V or V' does not belong to $[\alpha]$ then $\mathcal{A}$ aborts and returns a random bit. If $\mathcal{A}$ has not yet aborted, it generates two $1 \times \eta$ vectors $\overrightarrow{V} = (\overrightarrow{v}_1, \overrightarrow{v}_2, \ldots, \overrightarrow{v}_\eta)$ and $\overrightarrow{V}' = (\overrightarrow{v}'_1, \overrightarrow{v}'_2, \ldots, \overrightarrow{v}'_\eta)$, such that $\overrightarrow{v}_i \neq \overrightarrow{v}'_i$ for any $i \in [1, \eta], \overrightarrow{v}_i, \overrightarrow{v}_i \in [\alpha], \forall i \in [1, \eta]$ and $\sum_{i=1}^{\eta} \overrightarrow{v}_i = \sum_{i=1}^{\eta} \overrightarrow{v}'_i$. $\mathcal{A}$ returns $(\overrightarrow{V}, \overrightarrow{V}')$. In return $\mathcal{A}$ receives the challenge $D_b = (c_1^b, c_2^b, \ldots, c_\eta^b), \mathbb{X} = (X_1, X_2, \ldots, X_\eta), s = \sum_{i=1}^{\eta} \log_{\tilde{g}}(X_i)$. Note that, here $c_i^b \in \{g^{v_{1i}} * DH_{\tilde{g}}(g, X_i), g^{v'_{1i}} * DH_{\tilde{g}}(g, X_i)\}$. $\mathcal{A}$ selects $\overrightarrow{s} \in_R \mathbb{Z}_p^n$. Let us assume $\overrightarrow{s} = (\overrightarrow{s}_1, \overrightarrow{s}_2, \ldots, \overrightarrow{s}_n)$. Now, $\mathcal{A}$ generates two $n \times (\eta - 1)$ matrices $\mathbf{K} = [k_{ij}]$ and, $\mathbf{L} = [l_{ij}]$, such that $k_{ij} = (v_{ij} - v'_{ij}) * (v_j - v'_j)^{-1}$, and $l_{ij} = (v_{ij} * v'_j - v'_{ij} * v_j) * (v'_j - v_j)^{-1}$ for all $i \in [1, n], j \in [1, \eta - 1]$. Since, $v_{ij}, v'_{ij}, v_j, v'_j \in \mathbb{Z}_p$ for all $i \in [1, n], j \in [1, \eta - 1]$ and $v_j \neq v'_j$ for all $j \in [1, \eta - 1]$, k_{ij} and l_{ij} can be computed for all $i \in [1, n], j \in [1, \eta - 1]$. Now, $\mathcal{A}$ computes

$$X_{ij} = X_i^{k_{ij}} * \tilde{g}^{l_{ij}} : \forall i \in [1, n], j \in [1, \eta - 1]$$

$$c_{ij}^b = (c_i^b)^{k_{ij}} * \tilde{g}^{l_{ij}} : \forall i \in [1, n], j \in [1, \eta - 1]$$

It is easy to see that if $c_i^b = g^{v_i} * DH_{\tilde{g}}(g, X_i)$, then $c_{ij}^b = g^{v_{ij}} * DH_{\tilde{g}}(g, X_{ij})$ for all $i \in [1, n], j \in [1, \eta - 1]$. Alternatively, if $c_i^b = g^{v'_i} * DH_{\tilde{g}}(g, X_i)$, then $c_{ij}^b = g^{v'_{ij}} * DH_{\tilde{g}}(g, X_{ij})$ for all $i \in [1, n], j \in [1, \eta - 1]$. Then $\mathcal{A}$ generates, $\forall i \in [1, n]$:

$$X_{i\eta} = \frac{\tilde{g}^{\overrightarrow{s}_i}}{\prod_{j=1}^{\eta-1} X_{ij}}$$

$$c_{i\eta}^b = \frac{g^{\overrightarrow{s}_i} * g^{\Delta_i}}{\prod_{j=1}^{\eta-1} c_{ij}^b}$$

Let us denote $X = [X_{ij}], D_b = [c_{ij}^b]$. Now, $\mathcal{A}$ invokes $\mathcal{B}$ with the following inputs: $g, \tilde{g}, \overrightarrow{s}, X, D_b, V, V'$. $\mathcal{B}$ returns a bit b'. $\mathcal{A}$ returns the same bit. Let us now calculate the success probability of $\mathcal{A}$. $Pr[Exp^G_{\mathcal{A},SDDH}(\lambda) = 1] = Pr[Exp^G_{\mathcal{A},SDDH}(\lambda) = 1 \big| \eta = \eta'] Pr[\eta = \eta'] + Pr[Exp^G_{\mathcal{A},SDDH}(\lambda) = 1 \big| \eta \neq \eta'] Pr[\eta \neq \eta'] = Pr[Exp^G_{\mathcal{B},LDDH}(\lambda) = 1] * \frac{1}{poly(\lambda)} + (1 - \frac{1}{poly(\lambda)}) * \frac{1}{2} = \frac{1}{2} + \frac{1}{poly(\lambda)} \left(Pr[Exp^G_{\mathcal{A},LDDH}(\lambda) = 1] - \frac{1}{2}\right) = \frac{1}{2} + \frac{1}{poly(\lambda)} Adv^G_{\mathcal{B},LDDH}(\lambda)$. Thus, $Adv^G_{\mathcal{A},SDDH}(\lambda) \geq Pr[Exp^G_{\mathcal{A},SDDH}(\lambda) = 1] - \frac{1}{2} = \frac{1}{poly(\lambda)} Adv^G_{\mathcal{B},LDDH}(\lambda)$. Hence, the result holds.

Now, we show that DRE-CV is secure against the $IND - CVA$ attack described in section 3. We consider a PPT adversary $\mathcal{B}$ who along with the challenger emulates the security experiment $Exp^G_{\mathcal{B},IND-CVA}(\lambda)$. First, the $Setup()$

function generates the public parameters, and other parameters including the maximum number of voters. Then the challenger invokes $\mathcal{B}$ with these parameters. $\mathcal{B}$ returns the number of honest voters η and two sets $V^0_{c\times\eta}$ and $V^1_{c\times\eta}$. The challenger flips a coin and depending upon the outcome, chooses a bit b. The set $V^b_{c\times\eta}$ will be used by the challenger as the set of confirmed votes for η honest voters. That is, each column of $V^b_{c\times\eta}$ will be the used as the confirmed vote of one of the η honest voters. Note that the audited ballots of honest voters are not chosen by $\mathcal{B}$. This is because the honest voters are totally controlled by the challenger. It is the prerogative of the challenger to decide how many audits an honest user may conduct. The challenger also lets the adversary decide the sequence in which the colluding voters and the honest voters arrive. All the confirmed and audited ballots are posted on the Bulletin Board along with the NIZK proofs of well-formedness(if any). Once, the polling is complete, the challenger posts the tally and the aggregate randomness on the Bulletin Board the adversary needs to predict the bit b. The experiment will be successful if the adversary can predict b correctly.

The following lemma shows that our scheme is secure against the $IND-CVA$ attack.

Lemma 6.

$$Adv^G_{\mathcal{B},IND-CVA}(\lambda) \leq Adv^G_{\mathcal{A},LDDH}(\lambda) + O\left(Adv^G_{\mathcal{G},UZK}(\lambda)\right)$$

Proof. We show that if there exists an adversary $\mathcal{B}$, against the $Exp^G_{\mathcal{B},IND-CVA}(\lambda)$, it could be used in the construction of another adversary $\mathcal{A}$, against the security experiment $Exp^G_{\mathcal{A},LDDH}(\lambda)$. $\mathcal{A}$ runs $\mathcal{B}$ as a subroutine. When $\mathcal{A}$ is invoked with the inputs $c, \alpha, g, \tilde{g}$, $\mathcal{A}$ selects $n \in poly(\lambda)$, and invokes $\mathcal{B}^G(c, n, \alpha, g, \tilde{g})$. $\mathcal{B}$ returns $(\eta, V^0_{c\times\eta}, V^1_{c\times\eta})$. $V^0_{c\times\eta}$, and $V^1_{c\times\eta}$ represent the two possible sets of votes of honest voters. $\mathcal{A}$ returns $(\eta, V^0_{c\times\eta}, V^1_{c\times\eta})$. If $Exp^G_{\mathcal{A},LDDH}(\lambda)$ is not aborted, $\mathcal{A}$ receives $s = (s_1, s_2, \ldots, s_c), X_{c\times\eta} = [X_{ij}]$ and the challenge $D_b = [c^b_{ij}]$, where $i \in [1, c], j \in [1, \eta]$. Now, $\mathcal{A}$ starts the polling process. The adversary $\mathcal{B}$ casts votes on behalf of all the colluding voters. $\mathcal{B}$ confirms/audits her ballots as many times she wants. Whenever $\mathcal{B}$ tries to cast/audit a ballot, the challenger($\mathcal{A}$) interacts with her as usual, generating the randomnesses and the ballots in accordance with the scheme. When the turn of the i'th honest voter comes, the DRE machine selects a column i of c^b and X. Now, let the confirmed ballot of the honest voter i be $B_i = (B_{i1}, B_{i2}, \ldots, B_{in})$, where $B_{ij} = \langle b_{ij}, \tilde{x}_{ij}\rangle$. $\mathcal{A}$ assigns $b_{ij} = c^b_{ij}$, and $\tilde{x}_{ij} = X_{ij}$ for $j \in [1, c]$. $\mathcal{A}$ may also perform any number of audits on behalf of the honest voters. When $\mathcal{A}$ audits, she generates arbitrary votes and randomnesses and produces ballots accordingly. Note that, $\mathcal{A}$ knows when she is going to audit a ballot, so she can generate bespoke ballots against self-chosen randomness and votes for every audit she makes. $\mathcal{A}$ generates simulated NIZK proof of well-formedness for all confirmed ballots which she posts on the Bulletin Board along with the ballots. Now, when all the votes are cast, $\mathcal{A}$ posts on the Bulletin Board the tally and the randomnesses on the Bulletin Board. $\mathcal{A}$ can generate the tally by adding the votes of all the audited ballots, the confirmed

ballots of the adversary, and the confirmed ballots of the honest voters which are $\sum_{j=1}^{\eta} v_{ij}$ as shown in the description of $Exp^G_{\mathcal{A},LDDH}(\lambda)$. Note that $V^b_{c\times\eta}$ is known to $\mathcal{A}$ in $Exp^G_{\mathcal{A},LDDH}(\lambda)$. $\mathcal{A}$ also receives the aggregate randomness vector $s = (s_1, s_2, \ldots, s_n)$ from the challenger of $Exp^G_{\mathcal{A},LDDH}(\lambda)$. She adds to s, the randomnesses produced by herself during the process of generating the ballots for adversary-controlled voters and the audited ballots of honest users. Thus, she computes the sum of all randomnesses of all the ballots generated during the voting process. Once $\mathcal{A}$ has posted this tally and the overall randomness on the Bulletin Board, the adversary $\mathcal{B}$ will return a bit b'. $\mathcal{A}$ can return the same bit.

Let us now calculate the success probability of $\mathcal{A}$. The difference between the security game $Exp^G_{\mathcal{B},IND-CVA}(\lambda)$, and the real world $IND-CVA$ experiment is that in $Exp^G_{\mathcal{B},IND-CVA}(\lambda)$ some of the NIZK proofs are simulated, where in real life they all of them will be computed by the prover(s). Let, $\mathcal{B}'$ be an adversary against $Exp^G_{\mathcal{B}',IND-CVA}(\lambda)$ that receives simulated proofs. Then, $\left|Adv^G_{\mathcal{B},IND-CVA}(\lambda) - Adv^G_{\mathcal{B}',IND-CVA}(\lambda)\right| \leq O\left(Adv^G_{\mathcal{G},UZK}(\lambda)\right)$. Hence, $Adv^G_{\mathcal{B},IND-CVA}(\lambda) - O\left(Adv^G_{\mathcal{G},UZK}(\lambda)\right) \leq Adv^G_{\mathcal{B}',IND-CVA}(\lambda) \leq Adv^G_{\mathcal{A},LDDH}(\lambda)$. Therefore, $Adv^G_{\mathcal{B},IND-CVA}(\lambda) \leq Adv^G_{\mathcal{A},LDDH}(\lambda) + O\left(Adv^G_{\mathcal{G},UZK}(\lambda)\right)$.

Now, we can unify all the results proven in this section and claim the following.

$$Adv^G_{\mathcal{B},IND-CVA}(\lambda) \leq poly(\lambda) * Adv^G_{\mathcal{A},DDH}(\lambda) + O\left(Adv^G_{\mathcal{G},UZK}(\lambda)\right)$$

Hence, our cumulative e-voting scheme will be secure against the chosen votes attack if the Decisional Diffie Hellman assumption holds in the group G, and if the proof systems used in the scheme are indeed zero knowledge.

The result of Lemma 6 shows that the attacker can learn no extra information that the tally cannot allow her to obtain. This result can be easily extended to show that if the attacker gains momentary access to the DRE machine while the polling is going on, she can only be able to learn the partial tally from the beginning of the polling till the point of gaining access, and as such she will also be able to compute the partial tally from the point of gaining access till the end of polling. The proof of this fact is almost similar to that of Lemma 6. Formally, if the polling begins at time $t = t_0$, and $t = t_e$, and the attacker gains access to the DRE machine at time $t = t_1, t_2, \ldots, t_{e-1}$, then the adversary will only be able to compute the partial tally of all votes cast between time $[t_{i-1}, t_i]$, for all $i = 1, 2, \ldots, k$. The DRE machine does not store the trapdoors, hence the attacker can never decrypt the individual votes cast between any two successive time points of accessing the DRE machine.

8 Overhead

In this section, we discuss the efficiency of our scheme. We measure the efficiency in terms of both computational and communication cost. In order to

generate a ballot, the DRE needs to do $2c$ exponentiations. The NIZK proof $\pi_{ij}\left[x_{ij} : B_{ij}, \tilde{X}_{ij}\right]$ requires $4a + 2$ exponentiations for each $j \in [1, c]$, thus, for all $j \in [1, c]$ the figure sums up to $4ac + 2c$. The NIZK proof $\pi_i\left[x_i : B_i, \tilde{X}_i\right]$ requires 2 exponentiations. Thus, the total number of exponentiations required per confirmed ballot is given by $4ac + 4c + 2$. The audited ballots do not have NIZK proofs, hence, each of them requires only $2c$ exponentiations to be computed. If there are β audited ballots, the total computation cost will be $(4ac + 4c + 2)n + 2c\beta$.

The size of a ballot is $2c$. The size of the $c + 1$ NIZK proofs associated with each ballot is $4(a+1)c+4$. Thus, the overall communication cost of transferring each confirmed ballot to the Bulletin Board is $4ac+6c+4$. The size of the secret randomness associated with each audited ballot is c. So, the size of each of the audited ballot is $3c$. If there are n confirmed ballots and β audited ballots, the total communication cost will be $(4ac + 6c + 4)n + 3c\beta$.

	Computation Cost				Communication Cost			
Type	Ballot	Secret Key	NIZKP	Total	Ballot	Secret Key	NIZKP	Total
Audited	$2c$	0	–	$2c$	$2c$	c	–	$3c$
Confirmed	$2c$	0	$4ac + 2c + 2$	$4ac + 4c + 2$	$2c$	-	$4(a + 1)c + 4$	$4ac + 6c + 4$

Table 1. The computation and communication cost for the proposed cumulative voting scheme.

9 Related Work

Now, we shall discuss the existing works on formal definitions of privacy and verifiability of e-voting systems.

9.1 Verifiability

In this section, we discuss related previous works on formalization of verifiability notion in e-voting systems. Küsters et al. proposed a general definition of accountability in [32]. Their proposed KTV framework encompasses the notion of verifiability. In this definition there is a judge that accepts a run of a protocol if a certain goal is met. They call a protocol verifiable if there is a tolerance limit $\delta \in [0, 1]$, such that the judge will accept the run of the protocol that does not meet the goal γ with a probability not more than δ.

The first formal definition of verifiability was proposed by Benaloh in [9]. They consider an l-threshold, m teller and n voter election system. According to their definition, an (l, m, n) election schema is said to be verifiable, if the

probability that in an election system the checks return good and the tally is not correct is a negligible function of the number of proper tellers in that election system. One limitation of this definition is that they consider that the election is conducted on trusted machine which is a démodé assumption nowadays.

Kiayias et al. [1, 30] defined end-to-end verifiable e-voting scheme to be a five-tuple (Setup, Cast, Tally, Result, Verify) where each tuple is a probabilistic polynomial time algorithm. A total of 4 entities are involved in the election process: the voter, the election authority, the tellers and the Bulletin Board. The voters use a voter supporting device to cast their votes. Voters who have successfully cast their votes obtain a receipt. According to this definition, an e-voting system is end-to-end verifiable if at least θ voters have successfully terminated and the result does not deviate from the actual aggregate of all cast votes by more than k, where θ and k are parameters related to the definition of verifiability of the system.

Cortier et al. [20] proposed another definition of verifiable e-voting system. According to their definition, an e-voting system is a 7 tuple (Setup, Credential,Vote, VerifyVote, Valid, Board, Tally, Verify). All of them are probabilistic polynomial time algorithms. They considered verifiability against a malicious Bulletin Board, verifiability against a malicious registrar. An election protocol that is verifiable against both a malicious registrar and the malicious Bulletin Board is said to satisfy the strong verifiability property. On the other hand, for weak verifiability both the registrar and the Bulletin Board are assumed to be honest. Apart from these, they also introduced the notion of tally uniqueness. Here, the requirement is to ensure that there cannot be more than one result corresponding to the same run of the election protocol.

Smyth et al. [41] model an election scheme as a 4-tuple (Setup, Vote, Tally, Verify) of PPT algorithms. The algorithm vote is run by the voters and the algorithm Tally is run by the tellers. They proposed that an election system is verifiable if it satisfies both individual verifiability as well as universal verifiability. The election scheme is individually verifiable if the ballots of distinct voters are different with very high probability. An election protocol is universally verifiable if the adversary cannot simulate a run of the e-voting protocol, such that the tally function yields a different tally to the one which is output by the adversary and which passes the Verify test.

Baum et al. [4] presented a definition of game based correctness. They focussed on publicly auditable secure multi-party computation protocol in a UC framework where the input parties are different from working parties. [19] provided another definition according to which a voting scheme is verifiable if the tally is unique and the voting authority can find a witness that proves the correctness of the tally. Yet another definition of verifiability was provided by Szepieniec et al. in [42]. According to this definition a protocol P is verifiable if there exists a verifier V that can distinguish between the transcript of an honestly executed protocol from the one where some of the participants acted dishonestly.

A comparative study of all these works can be found in [21]. The difference between our verifiability definition and the ones proposed in the literature lies

in the fact that our scheme does not require a public, private key-pair of the election authority for generating encrypted ballots. Our verifiability definitions are crafted in a way so that the adversary controlling the DRE machine would not be able to deceive a voter with any significant probability if the requirements are met. In the case that the DRE machine is completely compromised, the tallying integrity will still be preserved due to the end-to-end verifiability and the information leakage will be limited to only the partial tally at the time of comprise. Our formal definitions of verifiability is satisfied without involving tallying authorities, which is different from the previous works.

9.2 Privacy

We now discussed existing works on game based definitions on privacy in e-voting techniques. The first game based definitions of privacy focused on unlinkability of votes to voters. Ideally, if there are two voters u_0, and u_1, and two votes v_0, and v_1, then the adversary will not be able to distinguish between the following two facts:

- u_0 chooses v_0, and u_1 chooses v_1 as her vote.
- u_0 chooses v_1, and u_1 chooses v_0 as her vote.

This type of definition can be found in [7,8]. One limitation of this definition is that it only considers swapping of votes, but does not consider different voting patterns that lead to the same tally. In order to avoid this shortcoming, Benaloh et al. proposed a new definition of privacy in [9]. [11, 12, 14] proposed another definition that says the adversary should not be able to distinguish between a real and a fake Bulletin Board that does not show the tally corresponding to the fake Bulletin Board. A variation of this definition was proposed in [23] that discusses ballot privacy against computationally unbounded adversary. For this purpose they proposed a new primitive called commitment consistent encryption (CCE), that can be used alongside a voting scheme. This primitive makes it possible to obtain a verifiable elections with a perfectly private audit trail which preserves the privacy of the votes even against a computationally unbounded adversary. Another proposal was given in [13] in which the adversary is given access to the tally as well as the a proof of correctness of the tally. When the adversary is given a fake Bulletin Board, the challenger uses a simulator to create a simulated proof of correctness of the tally corresponding to the real board. Smyth et al. modified this definition in [40]. Chase et al. provided yet another definition of privacy in [15]. In this definition the adversary sees real tally and real NIZK proofs, however, the definition makes it necessary for the tally to be same in two Bulletin Boards the adversary needs to distinguish between. Cortier et al. provided yet another definition in [10]. [10] also provides a survey of all previous works on formal definition of privacy in e-voting.

Existing definitions of privacy generally rely on a tallying authority (constituted by one or more trustees) that owns a secret key to the decryption oracle of encrypted ballots. Ballots are encrypted using the corresponding public key.

A set of trustees then jointly compute the tally from these encrypted ballots. Existing privacy definitions are designed to follow this setting. However, in our scheme there is no such tallying authorities. The DRE machine does not need to store any keys to compute ballots corresponding to voters' choices. Instead the DRE machine generates a ballot depending upon the choice of the voter, and the current state of the DRE. Every time a ballot is generated, the state of the DRE machine changes. After the last ballot is generated, the final state of the DRE machine is posted on the bulletin board. The initial state of the DRE machine is known to all. Now, anyone can compute the tally from the information published on the bulletin board. It is worth nothing that in any DRE-based voting system, the DRE machine learns the vote by definition as the voter chooses a candidate on the touch screen (however, the machine does not know the identity of the voter since the logon credential is randomly assigned to each voter after authentication) [27] . Therefore, our definition of privacy focuses on a scenario where the adversary has no access to the DRE machine internal state, but can cast votes on behalf of corrupt voters, and then gets access to all the public data, including the ballots, the initial and final states of the DRE.

10 Conclusion

In this paper we seek to find new formal definitions for DRE-based verifiable e-voting systems without tallying authorities. We observe that the existing formal definitions of privacy and verifiability rely on employing trustworthy tallying authorities for tallying votes. We have proposed a new framework for E2E verifiable e-voting scheme that uses DRE systems without involving any tallying authorities. In addition, we have also proposed a new DRE-based cumulative e-voting scheme. We have proven that this new e-voting scheme is compatible with the definition of privacy and verifiability proposed in this paper. We have also analyzed the system complexity of this e-voting system.

Acknowledgement

Feng Hao and Samiran Bag would like to acknowledge the support by the Royal Society grant, ICA/R1/180226.

References

1. Demos-2: Scalable e2e verifiable elections without random oracles. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 352–363. ACM, 10 2015.
2. Ben Adida. Helios: Web-based open-audit voting. In *Proceedings of the 17th Conference on Security Symposium*, SS'08, pages 335–348, Berkeley, CA, USA, 2008. USENIX Association.

3. Ben Adida and Ronald L. Rivest. Scratch & vote: Self-contained paper-based cryptographic voting. In *Proceedings of the 5th ACM Workshop on Privacy in Electronic Society*, WPES '06, pages 29–40, New York, NY, USA, 2006. ACM.
4. Carsten Baum, Ivan Damgård, and Claudio Orlandi. Publicly auditable secure multi-party computation. In Michel Abdalla and Roberto De Prisco, editors, *Security and Cryptography for Networks*, pages 175–196, Cham, 2014. Springer International Publishing.
5. Susan Bell, Josh Benaloh, Michael D. Byrne, Dana Debeauvoir, Bryce Eakin, Philip Kortum, Neal McBurnett, Olivier Pereira, Philip B. Stark, Dan S. Wallach, Gail Fisher, Julian Montoya, Michelle Parker, and Michael Winn. Star-vote: A secure, transparent, auditable, and reliable voting system. In *2013 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 13)*, Washington, D.C., August 2013. USENIX Association.
6. Josh Benaloh. Simple verifiable elections. *EVT*, 6:5–5, 2006.
7. Josh Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *Proceedings of the Twenty-sixth Annual ACM Symposium on Theory of Computing*, STOC '94, pages 544–553, New York, NY, USA, 1994. ACM.
8. Josh C Benaloh and Moti Yung. Distributing the power of a government to enhance the privacy of voters. In *Proceedings of the Fifth Annual ACM Symposium on Principles of Distributed Computing*, PODC '86, pages 52–62, New York, NY, USA, 1986. ACM.
9. Josh Daniel Cohen Benaloh. *Verifiable Secret-ballot Elections*. PhD thesis, New Haven, CT, USA, 1987. AAI8809191.
10. D. Bernhard, V. Cortier, D. Galindo, O. Pereira, and B. Warinschi. Sok: A comprehensive analysis of game-based ballot privacy definitions. In *2015 IEEE Symposium on Security and Privacy*, pages 499–516, May 2015.
11. David Bernhard, Véronique Cortier, Olivier Pereira, Ben Smyth, and Bogdan Warinschi. Adapting helios for provable ballot privacy. In Vijay Atluri and Claudia Diaz, editors, *Computer Security – ESORICS 2011*, pages 335–354, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
12. David Bernhard, Vronique Cortier, Olivier Pereira, and Bogdan Warinschi. Measuring vote privacy, revisited. pages 941–952, 10 2012.
13. David Bernhard, Olivier Pereira, and Bogdan Warinschi. How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to helios. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, pages 626–643, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
14. David Bernhard, Olivier Pereira, and Bogdan Warinschi. On necessary and sufficient conditions for private ballot submission. *IACR Cryptology ePrint Archive*, 2012:236, 2012.
15. Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Verifiable elections that scale for free. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *Public-Key Cryptography – PKC 2013*, pages 479–496, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
16. D. Chaum, A. Essex, R. Carback, J. Clark, S. Popoveniuc, A. Sherman, and P. Vora. Scantegrity: End-to-end voter-verifiable optical- scan voting. *IEEE Security & Privacy*, 6(3):40–46, May 2008.
17. David Chaum. Blind signatures for untraceable payments. In *Advances in cryptology*, pages 199–203. Springer, 1983.
18. David Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE security & privacy*, 2(1):38–47, 2004.

19. Benoît Chevallier-Mames, Pierre-Alain Fouque, David Pointcheval, Julien Stern, and Jacques Traoré. *On Some Incompatible Properties of Voting Schemes*, pages 191–199. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.
20. Véronique Cortier, David Galindo, Stéphane Glondu, and Malika Izabachène. Election verifiability for helios under weaker trust assumptions. In *19th European Symposium on Research in Computer Security - Volume 8713*, ESORICS 2014, pages 327–344, New York, NY, USA, 2014. Springer-Verlag New York, Inc.
21. Véronique Cortier, David Galindo, Ralf Küsters, Johannes Müller, and Tomasz Truderung. Sok: Verifiability notions for e-voting protocols. *2016 IEEE Symposium on Security and Privacy (SP)*, pages 779–798, 2016.
22. Chris Culnane, Peter Y. A. Ryan, Steve Schneider, and Vanessa Teague. vvote: a verifiable voting system (DRAFT). *CoRR*, abs/1404.6822, 2014.
23. Édouard Cuvelier, Olivier Pereira, and Thomas Peters. Election verifiability or ballot privacy: Do we need to choose? In Jason Crampton, Sushil Jajodia, and Keith Mayes, editors, *Computer Security – ESORICS 2013*, pages 481–498, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
24. Kevin Fisher, Richard Carback, and Alan T. Sherman. Punchscan: Introduction and system definition of a high-integrity election system. In *Workshop on Trustworthy Election. 2006*, 2006.
25. Jens Groth. Simulation-sound nizk proofs for a practical language and constant size group signatures. In Xuejia Lai and Kefei Chen, editors, *Advances in Cryptology – ASIACRYPT 2006*, pages 444–459, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.
26. Feng Hao, Dylan Clarke, Brian Randell, and Siamak F Shahandashti. Verifiable classroom voting in practice. *IEEE Security & Privacy*, 16(1):72–81, 2018.
27. Feng Hao, Matthew N. Kreeger, Brian Randell, Dylan Clarke, Siamak F. Shahandashti, and Peter Hyun-Jeen Lee. Every vote counts: Ensuring integrity in large-scale electronic voting. In *2014 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 14)*, San Diego, CA, August 2014. USENIX Association.
28. Feng Hao and Peter YA Ryan. *Real-World Electronic Voting: Design, Analysis and Deployment*. CRC Press, 2016.
29. Dorota Kamrowska-Zaluska. Participatory budgeting in poland–missing link in urban regeneration process. *Procedia engineering*, 161:1996–2000, 2016.
30. Aggelos Kiayias, Thomas Zacharias, and Bingsheng Zhang. End-to-end verifiable elections in the standard model. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015*, pages 468–498, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.
31. R. Kusters, T. Truderung, and A. Vogt. Clash attacks on the verifiability of e-voting systems. In *2012 IEEE Symposium on Security and Privacy*, pages 395–409, May 2012.
32. Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Accountability: Definition and relationship to verifiability. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pages 526–535, New York, NY, USA, 2010. ACM.
33. C. Andrew Neff. Practical high certainty intent verification for encrypted votes, 2004.
34. Richard H Pildes and Kristen A Donoghue. Cumulative voting in the united states. *University of Chicago Legal Forum*, page 241, 1995.

35. P. Y. A. Ryan, D. Bismark, J. Heather, S. Schneider, and Z. Xia. Prêt á voter voter: a voter-verifiable voting system. *IEEE Transactions on Information Forensics and Security*, 4(4):662–673, Dec 2009.
36. Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme. In Louis C. Guillou and Jean-Jacques Quisquater, editors, *Advances in Cryptology — EUROCRYPT '95*, pages 393–403, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.
37. Daniel Sandler, Kyle Derr, and Dan S. Wallach. Votebox: A tamper-evident, verifiable electronic voting system. In *Proceedings of the 17th Conference on Security Symposium*, SS'08, pages 349–364, Berkeley, CA, USA, 2008. USENIX Association.
38. Siamak F Shahandashti. Electoral systems used around the world. In *Real-World Electronic Voting (Eds. Hao, Ryan)*, pages 93–118. CRC Press, 2016.
39. Siamak F Shahandashti and Feng Hao. Dre-ip: a verifiable e-voting scheme without tallying authorities. In *European Symposium on Research in Computer Security*, pages 223–240. Springer, 2016.
40. Ben Smyth and David Bernhard. Ballot secrecy and ballot independence coincide. In Jason Crampton, Sushil Jajodia, and Keith Mayes, editors, *Computer Security – ESORICS 2013*, pages 463–480, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
41. Ben Smyth, Steven Frink, and Michael R. Clarkson. Computational election verifiability: Definitions and an analysis of helios and jcj. *IACR Cryptology ePrint Archive*, 2015:233, 2015.
42. Alan Szepieniec and Bart Preneel. New techniques for electronic voting. *USENIX Journal of Election Technology and Systems (JETS)*, 2015.

11 Appendix

11.1 NIZK Proof

$\Pi_{ij}\left[x_{ij} : B_{ij}, \tilde{X}_{ij}\right]$: This NIZK proof proves that given $\tilde{X}_{ij} = \tilde{g}^{x_{ij}}$ and B_{ij}, $B_{ij} = g^{x_{ij}} g^{v_{ij}}$, where $v_{ij} \in [a]$. This is same as proving the below statement: $\sigma \equiv (B_{ij} = g^{x_{ij}} \wedge \tilde{X}_{ij} = \tilde{g}^{x_{ij}}) \vee (B_{ij} = g^{x_{ij}} g \wedge \tilde{X}_{ij} = \tilde{g}^{x_{ij}}) \vee \ldots \vee (B_{ij} = g^{x_{ij}} g^{a} \wedge \tilde{X}_{ij} = \tilde{g}^{x_{ij}})$. σ is a one-out-of-$(a+1)$ OR statement. Exactly one of the sub-statements can be true. Let us assume that $\kappa \in [a]$ be such that $B_{ij} = g^{x_{ij}} g^{\kappa}$. The prover chooses $res_k, ch_k \in_R \mathbb{Z}_p$, for $k \in [1, a+1] \setminus \kappa$ and computes $com_k = \tilde{g}^{res_k} \tilde{X}_{ij}^{ch_k}$, and $com'_k = g^{res_k} X_{ij}^{ch_k} : \forall j \in [1, a+1] \setminus \kappa$. The prover also chooses $r \in_R \mathbb{Z}_p$ and computes $com_\kappa = \tilde{g}^r$ and $com'_\kappa = g^r$. Now, let ch be the grand challenge of the NIZK proof. This challenge ch is obtained as the output of a hash function that takes as input all the public information, and commitments generated thus far. In order to defeat clash attack [31], we should also include the ballot id as an input to this hash function. The prover computes $ch_\kappa = ch - \sum_{k \in [1,a+1]\setminus\kappa} ch_k$. The prover computes $res_\kappa = r - ch_\kappa * x_{ij}$.

The verification equations are as below:

1. $\tilde{g}^{res_k} \stackrel{?}{=} \frac{com_k}{\tilde{X}_{ij}^{ch_k}} : k \in [1, a+1]$.
2. $g^{res_k} \stackrel{?}{=} \frac{com'_k}{(B_{ij}/g^{k-1})^{ch_k}} : k \in [1, a+1]$.
3. $ch = \sum_{k=1}^{a+1} ch_k$

If these $2(a+1)$ equations are satisfied, the NIZK proof is correct. The NIZK proof consists of $2(a+1)$ commitments, $a+1$ challenges, and $a+1$ responses. Hence, the size of the NIZK proof is $4(a+1)$. The prover needs to do $4a+2$ exponentiations, whereas the verifier needs to do $4(a+1)$ exponentiations.

$\Pi_i\left[x_i : B_i, \tilde{X}_i\right]$: This NIZK proof proves that $\sum_{j=1}^{c} v_{ij} = a$, given $B_i = (B_{i1}, B_{i2}, \ldots, B_{ic})$, and $\tilde{X}_i = (\tilde{X}_{i1}, \tilde{X}_{i2}, \ldots, \tilde{X}_{ic})$. Let, $B = \prod_{k=1}^{c} B_{ik}$ and $\tilde{X} = \prod_{k=1}^{c} \tilde{X}_{ik}$. Obviously, $B = \tilde{X} * g^a$. The prover(DRE) selects $r \in_R \mathbb{Z}_p$, and computes two commitments $com = \tilde{g}^r$ and $com' = g^r$. Let the challenge be ch. The prover generates a response $res = r - ch * (\sum_{k=1}^{c} x_{ik})$. The verification equations are as follows:

1. $\tilde{g}^{res} \stackrel{?}{=} \frac{com}{\tilde{X}^{ch}}$
2. $g^{res_k} \stackrel{?}{=} \frac{com'}{(B/g^a)^{ch}}$

This NIZK proof comprises two commitment, one challenge and one response, thus the total size of the NIZK proof is 4. The prover needs to do two exponentiations for generating the proof whereas, the verifier needs to do 4 exponentiations for verifying it.