End-to-End Verifiable Elections in the Standard Model[*]

Aggelos Kiayias[‡], Thomas Zacharias[†‡] and Bingsheng Zhang[‡]

Dept. of Informatics and Telecommunications,
National and Kapodistrian University of Athens, Greece
{aggelos, thzacharias, bzhang}@di.uoa.gr

April 17, 2015

Abstract

We present the cryptographic implementation of "DEMOS", a new e-voting system that is *end-to-end verifiable* in the standard model, i.e., without any additional "setup" assumption or access to a random oracle (RO). Previously known end-to-end verifiable e-voting systems required such additional assumptions (specifically, either the existence of a "randomness beacon" or were only shown secure in the RO model). In order to analyze our scheme, we also provide a modeling of end-to-end verifiability as well as privacy and receipt-freeness that encompasses previous definitions in the form of two concise attack games.

Our scheme satisfies end-to-end verifiability *information theoretically* in the standard model and privacy/receipt-freeness under a computational assumption (subexponential Decisional Diffie Helman). In our construction, we utilize a number of techniques used for the first time in the context of e-voting schemes that include utilizing randomness from bit-fixing sources, zero-knowledge proofs with imperfect verifier randomness and complexity leveraging.

1 Introduction

In an end-to-end (E2E) verifiable election system, voters have the ability to verify that their vote was properly cast, recorded and tallied into the election result. Intuitively, the security property that an E2E verifiable election intends to capture is the ability of the voters to detect a malicious *election authority* that tries to misrepresent the election outcome. E2E verifiability is a strong level of security for election systems that has been widely accepted as a fundamental requirement for their adoption, see e.g., [43].

In more details, E2E verifiability mandates that the voter can obtain a receipt at the end of the ballot casting procedure that can allow her to verify that her vote was (i) cast as intended, (ii) recorded as cast, and (iii) tallied as recorded. Furthermore, any external third party should be able to verify that the election procedure is executed properly. In fact, it is imperative that the receipts in an E2E system are delegatable i.e., the voter may delegate the task of verifiability to any interested third party, for instance an international organization of the voters' choosing that aggregates the task of verification. This requirement, as well as the fact that it should be infeasible for the voter to use her receipt as a proof of the way she voted (this is necessary to deter vote-selling/buying), make the design of end-to-end verifiable systems a challenging problem.

All known e-voting systems that offer E2E verifiability provide it under some setup assumption or in the random oracle (RO) model. Notably, E2E verifiability can be *argued* (note that it is never formally

[*]The authors are ordered alphabetically and the third author is the corresponding author.

[†]Research was supported by ERC project CODAMODA.

[‡]Research was supported by project FINER, Greek Secretariat of Research and Technology funded under action "ARISTEIA 1."

proven before) for Helios [1] in the RO model while for Remotegrity[1] [49] in the model where a trusted party (a "randomness beacon") provides a stream of unbiased and unpredictable random coins. More general approaches for defining auditable multiparty computation (MPC) have recently been proposed [3] and also rely on a setup assumption such as a CRS.

A critical shortcoming of using setup assumptions for establishing E2E verifiability in e-voting is the fact that the voters will be required to make a "leap of faith" and accept the setup assumption in order to accept the election result. This can be an unfortunate state of affairs: since the election authority (EA) cannot unequivocally convince the voters that the election is correct, then the election outcome can be always subject to dispute.

Our Results. Motivated by the above, we design a new e-voting system that we can prove E2E verifiable *information theoretically* in the standard model, i.e., without any setup assumption except the existence of a bulletin board (BB) which provides a consistent view of the election transcript. Our result is further strengthened by the fact that we make the absolute minimal assumptions on the computation capabilities of the voters: voters are merely modeled as finite state *transducers* and thus are incapable of performing any cryptographic operation during ballot-casting (note the auditing stage after the election would require the capability of cryptographic operations but they can be performed at any time, in the post-election stage).

To accomodate the analysis of our system we provide a model for E2E verifiability and voter privacy/receipt-freeness. Our model for E2E verifiability is inspired from input-indistinguishable computation of Micali, Pass and Rosen [39] since in their setting they are also faced with proving security for multiparty computation in the standard model (note however they do not deal with E2E verifiability/auditability in their setting). In our modeling, the election system involves three types of entities, the voters $V_1, \ldots, V_n$, the election authority (EA), and the bulletin board (BB) whose only role is to provide storage for the election transcript for the purpose of verification. Voters submit their votes by engaging in the ballot casting protocol to the EA and they are not allowed to interact with each other. Our definition of end-to-end verifiability considers a very powerful adversary that is computationally unbounded and *completely* controls the EA. On the other hand, BB is completely passive and is only writeable by the EA and readable by anyone. The definition is satisfied, if and only if the adversary is incapable of evading being detected when it manipulates the election result as long as a number of voters perform the verifiability procedure honestly. Voter privacy on the other hand, considers an adversary that has full access to all the the voters' receipts, views of the ballot casting protocol as well as it may control of a number of malicious voters. For any election tally, the adversary should be incapable of distinguishing the way honest voters vote.

Our construction cherry picks ideas put forth in previous works, specifically, code-voting and double ballots from [13, 14], but also introduces a number of novel elements that enable us to prove E2E verifiability in the standard model. In order to achieve verifiability, our system utilizes a novel ZK proof for candidate encoding correctness and collects coins from the voters to form the challenge (specifically, a single random coin per voter). Given that the majority of voters cannot be assumed to be properly following the protocol, the sequence of voter contributed randomness is a particularly "weak source" that cannot be used for arguing the integrity of the election in a direct way — as we argue it is a very weak source akin to adaptive bit-fixing sources [38]. We then show (i) how it is possible to perform our ZK proof with a verifier that has imperfect randomness (just a min-entropy source), (ii) how to produce a (sufficiently long) sequence of min-entropy challenge from the random bits contributed by the voters. The tools that are important in our construction include a generalization of the Schwartz-Zippel lemma [46, 50] for imperfect randomness and a suitable strategy for dividing the coins of the voters so that the entropy is not lost due to the adversarial strategy of the EA (who also controls a number of voters). Using these techniques we design a novel ZK protocol and we prove unconditionally end-to-end verifiability for our scheme. For voter privacy, we utilize complexity leveraging to construct a simulator that is capable of reducing a voter privacy attack to a subexponential DDH distinguisher and hence our

[1]Note that Remotegrity itself is only a "front-end" type of system. It will be E2E verifiable if combined with Scantegrity-II [15] as suggested by the authors of the paper.

system offers privacy and receipt-freeness under a computational assumption.

In summary, our e-voting system is the first construction achieving the properties E2E verifiability and voter privacy/receipt-freeness in the standard model. Furthermore, we prove E2E verifiability information theoretically assuming the voters are computationally restricted transducers that hence are incapable of performing any cryptographic operation during ballot casting. The only assumptions we make are subexponential Decisional Diffie Hellman assumption (for voter privacy/receipt-freeness) and a consistent bulletin board board. We remark that a consistent bulletin board can be easily seen to be a tight condition since without it, it is easy to verify that E2E verifiability of the election cannot be achieved: by controlling the BB, an adversarial EA can distribute voters to their own separate "islands" where within each one the voters will have their own verifiable view of an election result that can be - in reality - completely skewed. Implementing a consistent bulletin board is beyond our scope, however we note that it can be achieved in the standard model using Byzantine agrement (BA) (for BA, see e.g., [26]) by assuming secure channels between any pair of parties. In fact, recently, it is shown that one can achieve BA efficiently even without secure channels in a completely anonymous setting [27] hence removing the requirement for pairwise secure channels (but note that this latter work relies on proofs of work modeled in the RO model).

Why previously known techniques do not work. To motivate further our approach it is worth-while to emphasize in which way previous works fail to attain end-to-end verifiability in the standard model. Helios, culminates a long line of previous schemes that employ homomorphic type of voting, cf. [18, 23, 33] and utilizes the Benaloh challenge [5] as the fundamental mechanism to attain verifiability. Helios by design requires the voter to utilize a voter supporting device to prepare a ciphertext and after an indeterminate number of trials, the voter will cast the produced ciphertext. Such ciphertexts are to be homomorphically tallied and thus they should be accompanied by a proof of proper computation. While such proofs are easy to construct based on e.g., [21], they can only be argued interactively (which is insufficient in our setting since a corrupt EA together with a corrupt voter may cook up a malformed proof that is indistinguishable from a proper one) or using a NIZK [10]. This latter approach is taken in Helios where a RO-based NIZK is utilized. In case the RO is dropped in favor of a standard model NIZK, security would be impossible in our model as NIZK's require a common reference string (CRS) and this is unavailable in the standard model; if the CRS is setup by the EA then in case it is malicious it will know and exploit the trapdoor; on the other hand, the voters are not interacting with each other and hence cannot setup the CRS by employing an MPC protocol. It follows that obtaining E2E verifiability in the standard model is impossible to overcome for Helios or any other similar existing scheme. On the other hand, in the case of Remotegrity/Scantegrity n coins need to be obtained from the randomness beacon in order to prove the result correct. It is easy to verify that the system is insecure in terms of end-to-end verifiability in case the randomness beacon is biased. As before, the only parties active are the EA and the voters who cannot implement a randomness beacon that is required in the construction. In light of the above, our construction offers a new paradigm in e-voting design: the randomness for the verification of the election can be collected distributively from the voters. Given that such randomness is by nature very weak (humans are very bad "randomness generators" and even worse malicious voters may collaborate with the election authority to cancel the honest voters' random bits) we show how suitable cryptographic techniques that deal with imperfect randomness can be employed to prove security.

Distributing the Election Authority. In our security model, we consider the EA as a single entity that is malicious in the verifiability game and honest in the privacy game. In practice one may want to distribute the EA to a number of "trustees" that collectively implement the EA functionality to improve the resiliency of the privacy property. While this is not a prime focus of our work (which centers on verifiability), it is feasible to design an efficient threshold protocol for implementing the EA. Note that our notion of voter privacy and receipt freeness can be easily extended to allow corrupted sub-authorities.

Other required properties of election systems. Our work by no means solves the complete set of desired requirements that are needed in an election system. Our voter-privacy definition implies receipt-freeness, i.e., provided that the voter receives the voter secret-key over an untappable channel[2], the voter

[2]An untappable channel enables the voter to deny the information that was transmitted in it. Physically distributing voter's

cannot convince any third party about the way she voted. Nevertheless, this does not imply coercion resistance as the voter may still be forced to divulge the voter secret prior to her ballot-casting (this does not violate voter privacy - it just prevents the voter from actually using the system and enables the adversary to vote on the voter's behalf). There are techniques that can be used to increase coercion resistance for internet-voting (e.g., those of [31, 17] and others) and they are compatible with our construction. We leave the integration of these techniques with information theoretic E2E verifiability for future work. Similarly, usability aspects are not within our current scope; nevertheless, we stress that we have implemented our system for 1-out-of-m elections and we have used it in real-world experiments[3].

Related work. In [12], Chaum suggested for the first time that anonymous communication can lead to voting systems with *individual verifiability*, i.e., the voters can verify that their votes were counted correctly. In [45], Sako and Killian introduce explicitly the notion of *universal verifiability*, that is, the ability for anyone to verify that the election result derives from the cast votes. Universal verifiability is also defined by Juels, Catalano and Jakobsson in [31] in the computational model assuming a trusted setup. Kremer, Ryan and Smyth [34] introduced symbolic definitions for individual and universal verifiability in the context of applied pi calculus. A formal definition of universal verifiability is also provided by Chevallier-Mames *et al.* in [16].

End-to-end verifiability in the sense of cast-as-intended, recorded-as-cast, tallied-as-recorded was an outcome of the works of Chaum [14] and Neff [42]. The novelty was the generation of receipts that could be used for simple voter verification while achieving privacy. The term of E2E verifiability (more precisely, E2E integrity) also appeared in [20]. Marneffe, Pereira and Quisquater presented an ideal-world definition for election systems in [24] without explicitly considering verifiability as a property of the ideal world. In [43], Popoveniuc *et al.* proposed a definition of E2E verifiability via a list of properties.

Küsters, Truderung and Vogt [35] introduced symbolic and computational definitions of verifiability parameterized by a goal and an adversarial environment. In [37], the same authors showed that individual verifiability and universal verifiability are not sufficient to guarantee the "global" verifiability of an e-voting system. A number of other e-voting systems in the cryptographic setting that do not explicitly deal with E2E verifiability include [19, 7, 22, 23].

Benaloh and Fischer [19] provided a computational definition of privacy as the property that any coalition of malicious voters cannot distinguish between any two vote assignments coming from a subset of honest voters that have the same partial tally. Receipt-freeness has been first studied by Benaloh and Tuinstra [6] and described as the property of an e-voting system to generate fake voter transcripts that are indistinguishable from genuine transcripts. Following this logic, in our voter privacy/receipt-freenes definition, we require simulation-based indistiguishability of the views of the voters when they engage in the ballot-casting stage. Chevallier-Mames *et al.* [16] introduced definitions for unconditional of privacy and receipt-freeness and showed incompatibility results of universal verifiability with each of these two properties.

Formal definitions for privacy and receipt-freeness have been proposed in the context of applied pi calculus [25] and the universal composability model [29, 41]. In [37], the level of privacy of an e-voting system is measured w.r.t. to the observation power the adversary has in a protocol run, via a definition which is close to the Dolev-Yao model.

In [8], Bernhard *et al.* proposed a game-based notion of ballot privacy and study the privacy of Helios. In their model, an adversary that chooses a fixed vote E, cannot distinguish a bulletin board that contains ballots for real votes from a bulletin board that contains ballots for E. Their definition was extended by Bernhard, Pereira and Warinschi [9] by allowing the adversary to statically corrupt election authorities. Both these definitions, although they imply a strong inditinguishability property, do not consider receipt-freeness. We note that our game-based definition captures both privacy and receipt-freeness while restricted to a single EA (and it can easily be extended by including a set of trustees that the adversary may corrupt).

secrets or using non-committing encryption [4] achieves untappability.

[3]For more information check our web-site `http://www.demos-voting.org`

As we have mentioned previously, modelling coercion resistance is out of the scope of this work. We refer the reader to [31, 25, 48, 36] for formal definitions of coercion resistance in the cryptographic, symbolic and universal composability model.

Organization. In Section 2, we introduce the syntax and define the correctness, E2E verifibiality and voter privacy/receipt freeness of an e-voting system. In Section 3, we present at length the construction of our e-voting system, including a detailed description of all tools that are applied. In Section 4, we prove the E2E verifibiality and voter privacy/receipt freeness of our e-voting system in the security framework of Section 2. In Section 5, we provide an overview of the implementation of our system for 1-out-of-m elections along with election preparation time benchmarks.

2 E-voting Systems

2.1 Preliminaries

We use λ as the security parameter. Associated with an e-voting system, we also consider two other parameters, the number of voters n and number of candidates m which are both thought as polynomial functions of λ. Let Π be an e-voting system, where $\mathcal{P} = \{P_1, ..., P_m\}$ is the set of candidates and $\mathcal{V} = \{V_1, ..., V_n\}$ is the set of voters. We denote by $\mathcal{U} \subseteq 2^{\mathcal{P}}$ the collection of subsets of candidates that the voters are allowed to choose to vote for (which may include a "blank" option too). The candidate selection $\mathcal{U}_\ell$ of voter V_ℓ is an element in $\mathcal{U}$.

Let $\mathcal{P}^*$ be the set of vectors of candidate selections of arbitrary length. Let f be the *election evaluation function* from $\mathcal{P}^*$ to the set $\mathbb{Z}_+^m$ so that $f(\mathcal{U}_1, \ldots, \mathcal{U}_n)$ is equal to an m-vector whose i-th location is equal to the number of times P_i was chosen in the candidate selections $\mathcal{U}_1, \ldots, \mathcal{U}_n$.

The entities involved in an e-voting system Π, are the voters $V_1, \ldots, V_\ell$, the *election authority (EA)* and the *Bulletin Board (BB)*.

2.2 Syntax and Correctness

An e-voting system Π is a quintuple of algorithms and protocols $\langle \mathbf{Setup}, \mathbf{Cast}, \mathbf{Tally}, \mathbf{Result}, \mathbf{Verify} \rangle$ specified as follows:

- The algorithm $\mathbf{Setup}(1^\lambda, \mathcal{P}, \mathcal{V}, \mathcal{U})$ is executed by the EA and generates a master secret key msk, Π's public parameters Pub (which include $\mathcal{P}, \mathcal{V}, \mathcal{U}$) and the voters' secrets $s_1, \ldots, s_n$. EA has a state, st which is initialized as msk. In addition, it posts an initial public transcript $\tau = \mathrm{Pub}$ on the BB.

- The interactive protocol $\mathbf{Cast}$ is between three parties, the voter V_ℓ, the BB and the EA. V_ℓ has input $(\mathrm{Pub}, s_\ell, \mathcal{U}_\ell)$, EA has input msk and BB has input τ. EA updates its state st and BB updates the public transcript τ. Upon successful termination, the voter V_ℓ receives a receipt α_ℓ. We denote by $view_\ell$ the view of the voter V_ℓ in the protocol $\mathbf{Cast}$.

- The interactive protocol $\mathbf{Tally}$ with common input Pub is executed by the EA and the BB on inputs msk, τ respectively. Upon successful termination, the BB updates the public transcript τ.

- The algorithm $\mathbf{Result}(\tau)$ outputs the result R_τ for the election or returns $\perp$ in case such result is undefined.

- The algorithm $\mathbf{Verify}(\tau, \alpha)$ outputs a value in $\{0, 1\}$, where α is a voter receipt (that corresponds to the voter's output from the $\mathbf{Cast}$ protocol).

Remark. In many election systems, the EA is implemented by more than a single authority. This means that $\mathbf{Setup}$ might be a protocol executed by those parties (as opposed to a standalone algorithm). However, from the point of view of E2E verifiability (where the system is considered malicious as a whole) this is completely immaterial. Hence, for simplicity in the syntax above we consider EA a single entity. In our construction the EA may also be distributed. We defer the details for how this may be done to the full version of the paper.

Definition 1 (Correctness) *The e-voting system* Π *has (perfect)* correctness*, if for any honest execution of* Π *that results in a public transcript* τ *where the voters* $V_1, \ldots, V_n$ *cast votes for options* $\mathcal{U}_1, \ldots, \mathcal{U}_n$ *and received receipts* $\alpha_1, \ldots, \alpha_n$*, it holds that*

$$\mathbf{Result}(\tau) = f(\mathcal{U}_1, \ldots, \mathcal{U}_n) \textit{ and } \wedge_{\ell=1}^{n} (\mathbf{Verify}(\tau, \alpha_\ell) = 1).$$

2.3 E2E Verifiability

In order to define E2E verifiability formally, we introduce a suitable notation; given that candidate selections are elements of a set of m choices, we may encode them as m-bit strings, where the bit in the i-th position is 1 if and only if candidate P_i is selected. Further, we may aggregate the election results as the list with the number of votes each candidate has received, thus the output of the $\mathbf{Result}$ algorithm is a vector in $\mathbb{Z}_+^m$. In this case, a result is feasible if and only if the sum of all its coordinates is no greater than the number of voters.

In our formalization of the E2E verifiability, we postulate the existence of a *vote extractor* algorithm $\mathcal{E}$ (not necessarily running in polynomial-time) that explains the election transcript: namely, it receives input of the form (τ, A) where τ is an election transcript and $A = \{\alpha_\ell\}_{\ell \in \tilde{\mathcal{V}}}$ is a set of $\mathbf{Cast}$ protocol receipts. By $\tilde{\mathcal{V}}$, we denote the set of honest voters that voted successfully. Given such input, $\mathcal{E}$ will compute $n - |\tilde{\mathcal{V}}|$ vectors $\langle \mathcal{U}_\ell \rangle_{V_\ell \in \mathcal{V} \setminus \tilde{\mathcal{V}}}$ in $\{0,1\}^m$ (which correspond to the choices of all the voters outside of $\tilde{\mathcal{V}}$) that can be either a candidate selection if the voter has voted adversarially or a zero vector if the voter has not voted successfully; $\mathcal{E}$ returns the symbol $\perp$ in case such values cannot be defined. In the special case where all voters are honest and have voted successfully (i.e., $\tilde{\mathcal{V}} = \mathcal{V}$), $\mathcal{E}$ returns no value (outputs the empty set). The purpose of the $\mathcal{E}$ algorithm will be to capture the setting when the election transcript τ contains (in potentially encoded form) a set of well-formed actual votes.

Using the above notion, we will be capable to express the actual result encoded in an election transcript. Next, we want to formally express a measure of *deviation* from the actual election result (as such deviation is the objective of the adversary in an E2E verifiability attack). Some preliminary notions will be needed. In order to express formally the deviation the adversary aims at, it is natural to equip the space of results with a *metric*. We use the metric derived by the 1-norm, $\|\cdot\|_1$ scaled to half, i.e., $\mathrm{d}_1 : \mathbb{Z}_+^m \times \mathbb{Z}_+^m \longrightarrow \mathbb{R}$ with $\mathrm{d}_1(w, w') = \frac{1}{2} \cdot \|w - w'\|_1 = \frac{1}{2} \cdot \sum_{i=1}^{n} |w_i - w'_i|$ where w_i, w'_i is the i-th coordinate of w, w' respectively.

Consider $R \in \mathbb{Z}_+^m$ be the election results that correspond to the true voter intent of n voters, and $R' \in \mathbb{Z}_+^m$ be the published election results. Denote by $\max(\mathcal{U})$, the maximum cardinality of an element in $\mathcal{U}$. Two encodings of candidate selections are within $\max(\mathcal{U})$ distance, so intuitively, if the adversary wants to present R' as the result of the election, it may do that by manipulating the votes of at least $\mathrm{d}_1(R, R')/\max(\mathcal{U})$ voters.

We define next the E2E Verifiability game, $G_{\mathrm{E2E-Int}}^{\mathcal{A},\mathcal{E},d,\theta}$, between the adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ using a voter extractor $\mathcal{E}$, that takes as input the security parameter, λ, the number of candidates, m and the number of voters, n.

Overview of the game $G_{\mathrm{E2E-Ver}}^{\mathcal{A},\mathcal{E},d,\theta}(1^\lambda, m, n)$**.** The attack game is parameterized by d, which is the deviation amount (according to the metric $\mathrm{d}_1(\cdot,\cdot)$) that the adversary wants to achieve and θ, the minimum number of voters that $\mathcal{A}$ must allow to vote honestly and terminate successfully. The adversary starts by selecting the voter and candidate identities for given parameters n, m. It also determines the allowed ways to vote as described by the set $\mathcal{U}$. The adversary fully controls the EA. The adversary manages the $\mathbf{Cast}$ protocol executions where it assumes the role of the EA. For each voter, the adversary may choose to corrupt it or to allow the challenger to play on its behalf. In the second case, the adversary provides the candidate selection that the honest voter will use in the $\mathbf{Cast}$ protocol. The adversary completes the execution of EA which results to the complete election transcript published in the BB. The adversary will win the game provided that all θ honest voters that completed the **Cast** protocol successfully will also audit the result successfully but the deviation of the tally is at least d; the adversary will also win in case the extractor fails to produce the candidate selection of the dishonest voters (but θ honest voters still verify correctly). The attack game is specified in detail in Figure 1.

E2E Verifiability Game $G^{\mathcal{A},\mathcal{E},d,\theta}_{\mathrm{E2E-Ver}}(1^\lambda, m, n)$

1. $\mathcal{A}$ chooses a list of candidates $\mathcal{P} = \{P_1, ..., P_m\}$, a set of voters $\mathcal{V} = \{V_1, ..., V_n\}$ and the set of allowed candidate selections $\mathcal{U}$. It provides $\mathcal{C}$ with the sets $\mathcal{P}, \mathcal{V}, \mathcal{U}$ along with information Pub and voter credentials $\{s_\ell\}_{\ell\in[n]}$. Throughout the game, $\mathcal{C}$ plays the role of the BB.
2. The adversary $\mathcal{A}$ and the challenger $\mathcal{C}$ engages in an interaction where $\mathcal{A}$ schedules the **Cast** protocols of all voters. For each voter V_ℓ, $\mathcal{A}$ can either completely control the voter or allow $\mathcal{C}$ to operate on their behalf, in which case $\mathcal{A}$ provides a candidate selection $\mathcal{U}_\ell$ to $\mathcal{C}$. Then, $\mathcal{C}$ engages with the adversary $\mathcal{A}$ in the **Cast** protocol so that $\mathcal{A}$ plays the role of EA. Provided the protocol terminates successfully, $\mathcal{C}$ obtains the receipt α_ℓ on behalf of V_ℓ.
   Let $\tilde{\mathcal{V}}$ be the set of honest voters (i.e., those controlled by $\mathcal{C}$) that terminated successfully.
3. Finally, $\mathcal{A}$ posts the election transcript τ to the BB.

The game returns a bit which is 1 if and only if the following conditions hold true:

(i). $|\tilde{\mathcal{V}}| \geq \theta$, (i.e., at least θ honest voters terminated).

(ii). $\forall \ell \in [n]$: if $V_\ell \in \tilde{\mathcal{V}}$, then $\mathbf{Verify}(\tau, \alpha_\ell) = 1$ (i.e., the voters in $\tilde{\mathcal{V}}$ verify their ballot successfully).

and either one of the following two conditions:

(iii-a). If $\perp \neq \langle \mathcal{U}_\ell \rangle_{V_\ell \in \mathcal{V}\setminus\tilde{\mathcal{V}}} \leftarrow \mathcal{E}(\tau, \{\alpha_\ell\}_{V_\ell\in\tilde{\mathcal{V}}})$,
then
$$\mathrm{d}_1(\mathbf{Result}(\tau), f(\langle \mathcal{U}_1, \ldots, \mathcal{U}_n\rangle)) \geq d.$$

(iii-b). $\perp \leftarrow \mathcal{E}(\tau, \{\alpha_\ell\}_{V_\ell\in\tilde{\mathcal{V}}})$.

Figure 1: The E2E Verifiability Game between the challenger $\mathcal{C}$ and the adversary $\mathcal{A}$ using the vote extractor $\mathcal{E}$.

Definition 2 (E2E-Verifiability) *Let* $0 < \epsilon < 1$ *and* $n, m, d, \theta \in \mathbb{N}$ *with* $d > 0$ *and* $0 < \theta \leq n$. *The election protocol* Π *w.r.t. the election function* f *achieves* E2E verifiability *with error* ϵ, *for a number of at least* θ *honest successful voters and tally deviation* d *if there exists a (not necessarily polynomial-time) vote-extractor* $\mathcal{E}$ *such that for any adversary* $\mathcal{A}$:
$$\Pr[G^{\mathcal{A},\mathcal{E},d,\theta}_{\mathrm{E2E-Ver}}(1^\lambda, m, n) = 1] \leq \epsilon.$$

In plain words, Definition 2 suggests that an E2E verifiable e-voting system, provides an "official explanation" of adversarial votes via the vote extractor $\mathcal{E}$, such that if at least θ voters verify the result, then any adversary that attempts to manipulate the election tally (that includes the honest votes and the official explanation of the adversarial votes) by a shift of d votes will get caught except from some (supposedly small) probability ϵ.

Remark. In the only previous works [37, 35] where end-to-end verifiability was considered at a "global level" as we do here, it was expressed with respect to a set of "good" runs γ of the e-voting protocol in the sense that a judge could test whether the protocol operated within the set γ. Even though sufficiently expressive, this formulation has the disadvantage that the set γ remains undetermined and thus the level of verifiability that is offered by the definition hinges on the proper definition of γ which may not be simple. Using our language the notion of a good run becomes explicit: a run of the e-voting protocol is good provided that the extractor $\mathcal{E}$ produces votes for the malicious voters which if they are added to the votes of the honest voters they produce a result that does not deviate from the published result according to the $\mathrm{d}_1(\cdot,\cdot)$ metric. Note that our vote extractor may require super-polynomial time (in the same way that the set of good runs γ may have a membership test of super-polynomial complexity). We remark that the use of a super-polynomial extractor to define properly the inputs of the malicious participants and hence the soundness of a multiparty protocol is not novel to our work. For example see, Micali, Pass and Rosen [39] where they used a similar construct to prove security of their general multiparty computation protocol.

2.4 Voter Privacy (including Receipt-Freeness)

The definition of voter privacy concerns the actions that may be taken by the adversary to break the privacy and learn some information about the candidate selections of the honest voters. We specify the goal of the adversary in a very general way. In particular, for an attack against voter privacy to succeed, we ask that there is an election result, for which the adversary is capable of distinguishing how the honest voters voted while it has access to (i) the actual receipts that the voters obtained after ballot-casting as well as (ii) a set of protocol views that are consistent with all the honest voters' views in the $\mathbf{Cast}$ protocol instances they participated (and the adversary has observed).

Observe that any system that is secure against such an attack scenario would possess also "receipt-freeness", i.e., voters cannot prove how they voted by showing the receipt they obtain from the $\mathbf{Cast}$ protocol or even presenting their view in the $\mathbf{Cast}$ protocol. Given that in the privacy definition we allow the adversary to observe the view of the voter in the $\mathbf{Cast}$ protocol, we need to allow the voter to be able to "lie" about her view in this protocol (otherwise an attack could be trivially mounted). Note that this would require the voter input to the $\mathbf{Cast}$ protocol to be delivered via an untappable channel; in particular, the adversary should not have any side-channel information about the voter's secrets $s_1, \ldots, s_n$.

We formally define the voter privacy of an election via a *Voter Privacy/Receipt-freeness* game, denoted by $G^{\mathcal{A},\mathcal{S}}_{t\text{-priv}}$, that is played between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, that takes as input the security parameter, λ, the number of voters, n, and the number of candidates, m, as described in Figure 2 and returns 1 or 0 depending on whether the adversary wins. An important feature of the game is the existence of an efficient "voter simulator" $\mathcal{S}$ that provides a simulated view of the voter in the $\mathbf{Cast}$ protocol. Intuitively, this simulator captures the way the voter can lie about her candidate selection in the $\mathbf{Cast}$ protocol in case she is coerced to present her view after she completes the ballot-casting procedure.

Overview of the game $G^{\mathcal{A},\mathcal{S}}_{t\text{-priv}}(1^\lambda, n, m)$. The adversary starts by selecting the voters and candidates for given parameters n, m. It also determines the allowed ways to vote. The challenger flips a coin b (that will change its behavior during the course of the game) and will perform the $\mathbf{Setup}$ protocol. Subsequently, the adversary will schedule all $\mathbf{Cast}$ protocols selecting which voters it prefers to corrupt and which ones it prefers to allow to vote honestly. The adversary is allowed to corrupt at most t voters. The voters that remain uncorrupted are operated by the challenger and they are given two candidate selections to choose. The challenger will select which of the two candidate selections the voter will use in the $\mathbf{Cast}$ protocol according to the bit b. The adversary will also receive the receipt that is obtained by each voter as well as either the actual view of each voter during the $\mathbf{Cast}$ protocol, if $b = 0$, or a *simulated* view, if $b = 1$ (this addresses the receipt-freeness aspect). Upon completion of ballot-casting, the challenger executes the $\mathbf{Tally}$ protocol and posts the election result. Subsequently the adversary will attempt to guess b. The attack is successful provided that the adversary has corrupted up to t voters, the election tally is the same with respect to the two alternatives provided for each honest voter by the adversary and the adversary manages to guess the challenger's bit b. The game is presented in more detail in figure 2.

Definition 3 (Voter Privacy/Receipt-Freeness) *Let $n, m \in \mathbb{N}$. The e-voting system Π w.r.t. the election function f achieves* voter privacy/receipt-freeness *for at most t corrupted voters, if there is a PPT*

Voter Privacy/Receipt-freeness Game $G_{t\text{-priv}}^{\mathcal{A},\mathcal{S}}(1^\lambda, n, m)$

1. $\mathcal{A}$ on input $1^\lambda, n, m$, chooses a list of candidates $\mathcal{P} = \{P_1, ..., P_m\}$, a set of voters $\mathcal{V} = \{V_1, ..., V_n\}$, and the set of allowed candidate selections $\mathcal{U}$. It provides $\mathcal{C}$ the sets $\mathcal{P}$, $\mathcal{V}$, and $\mathcal{U}$.
2. $\mathcal{C}$ flips a coin $b \in \{0, 1\}$ and performs the **Setup** protocol on input $(1^\lambda, \mathcal{P}, \mathcal{V}, \mathcal{U})$ to obtain $msk, s_1, \ldots, s_n$, Pub; it provides $\mathcal{A}$ with Pub.
3. The adversary $\mathcal{A}$ and the challenger $\mathcal{C}$ engage in an interaction where $\mathcal{A}$ schedules the **Cast** protocols of all voters which may run concurrently. For each voter $V_\ell \in \mathcal{V}$, the adversary chooses whether V_ℓ is corrupted:
   - If V_ℓ is corrupted, then $\mathcal{C}$ provides s_ℓ to $\mathcal{A}$, and then they engage in a **Cast** protocol where $\mathcal{A}$ plays the role of V_ℓ and $\mathcal{C}$ plays the role of EA and BB.
   - If V_ℓ is not corrupted, $\mathcal{A}$ provides two candidate selections $\langle \mathcal{U}_\ell^0, \mathcal{U}_\ell^1 \rangle$ to the challenger $\mathcal{C}$. $\mathcal{C}$ operates on V_ℓ's behalf, using $\mathcal{U}_\ell^b$ as the V_ℓ's input. The adversary $\mathcal{A}$ is allowed to observe the network trace of the **Cast** protocol where $\mathcal{C}$ plays the roles of V_ℓ, EA, and BB. When the **Cast** protocol terminates, the challenger $\mathcal{C}$ provides to $\mathcal{A}$: (i) the receipt α_ℓ that V_ℓ obtains from the protocol, and (ii) if $b = 0$, the current view of the internal state of the voter V_ℓ, $view_\ell$, that the challenger obtains from the **Cast** execution, or if $b = 1$, a simulated view of the internal state of V_ℓ produced by $\mathcal{S}(view_\ell)$.
4. $\mathcal{C}$ performs the **Tally** protocol playing the role of EA and BB. $\mathcal{A}$ is allowed to observe the network trace of that protocol.
5. Finally, $\mathcal{A}$ using all information collected above (including the contents of the BB) outputs a bit b^*.

Denote the set of corrupted voters as $\mathcal{V}_{\mathsf{corr}}$ and the set of honest voters as $\tilde{\mathcal{V}} = \mathcal{V} \setminus \mathcal{V}_{\mathsf{corr}}$. The game returns a bit which is 1 if and only if the following hold true:

(i). $b = b^*$ (i.e., the adversary guesses b correctly).

(ii). $|\mathcal{V}_{\mathsf{corr}}| \leq t$ (i.e., the number of corrupted voters is bounded by t).

(iii). $f(\langle \mathcal{U}_\ell^0 \rangle_{V_\ell \in \tilde{\mathcal{V}}}) = f(\langle \mathcal{U}_\ell^1 \rangle_{V_\ell \in \tilde{\mathcal{V}}})$ (i.e., the election result w.r.t. the set of voters $\tilde{\mathcal{V}}$ does not leak b).

Figure 2: The Voter-privacy/Receipt-freeness game

voter simulator $\mathcal{S}$ such that for any PPT adversary $\mathcal{A}$:

$$\left| \Pr[G_{t\text{-priv}}^{\mathcal{A},\mathcal{S}}(1^\lambda, n, m) = 1] - 1/2 \right| = \mathsf{negl}(\lambda).$$

Remark. Our game-based voter privacy/receipt-freeness definition is close in spirit to witness indistinguishability of interactive proof systems. Namely, the adversary's challenge is to distinguish between two possible lists of candidate selections (the witnesses) that produce the same tally when restricted to just the honest voters. A potentially stronger privacy requirement would be a simulation-based formulation (akin to zero-knowledge in interactive proof systems) e.g., as the one suggested for ballot privacy in [9]. The definition of [9] is incomparable to ours because even though it is simulation-based and it captures malicious behavior of a subset of multiple trustees, it does not consider receipt-freeness. Note that our definition can be easily extended to the setting of multiple trustees (that the adversary may corrupt up to a threshold).

3 Presentation of Our e-Voting System

Our system has three stages, setup, ballot-casting and tallying, that parallel the operation of a Σ protocol. During setup stage, the EA produces a series of commitments and pre-audit data that correspond to a first move of a Σ protocol that will establish the validity of the commitments. During ballot-casting, voters engage with the EA in a protocol that will result in the recording of their votes, as well as in the submission of a random coin flip that will be used to produce the challenge for the Σ protocol. Voters will receive a receipt as their local output from the ballot-casting protocol that can be used for auditing the election result. In the third and final stage, the EA produces the tally of the election and completes the Σ protocol by publishing openings to commitments as well as other necessary information needed for verification. The verification step can take place at any time after the completion of the process using a collection of at least one receipt from the ballot-casting stage.

In our system, the voter implementation during the ballot-casting stage is expressed as a probabilistic transducer (see e.g., [30]) with a communication tape that has a number of states polynomial in the number of candidates m (and independent to other parameters such as n, λ). Given that such a machine is severely limited in the computational sense, in order to achieve ballot casting we utilize a code-voting approach (cf. [13]): the EA corresponds vote-codes to commitments posted in the BB, and voters cast their vote by simply sending to the EA the vote-code that they prefer. The commitments have an additive homomorphic property, hence it is possible to tally the result by homomorphically processing them and opening the resulting "tally commitment". The proof that we use in order to ensure verifiability is a conjunction of a cut-and-choose proof with a Σ proof that a committed value belongs to a set. The challenge needed for the Σ proof will be extracted by applying a suitable extraction mechanism to the coin flips of the voter transducers that are collected by the EA.

In Sections 3.1, 3.2, 3.3 and 3.4, we provide a detailed description of the tools that we apply for the construction of our system, i.e., (i) the homomorphic commitment scheme, (ii) a generalization of the Schwartz-Zippel lemma for imperfect randomness, (iii) the Σ protocol and (iv) the challenge extraction mechanism, respectively. We describe our e-voting system in Section 3.5 and prove its correctness in Section 3.6. For the better understanding of our system, we provide a toy example in Section 3.7.

3.1 Perfectly Binding Commitment

To achieve integrity against computationally unbounded adversaries, we have to use a perfectly binding commitment scheme. Moreover, our system requires such a commitment scheme to be additively homomorphic to facilitate the tally and audit process. In this work, we instantiate the commitment scheme with lifted ElGamal over elliptic curves. We use elliptic curve domain parameters $\mathsf{Param} := (p, a, b, g, q)$ generated by the curve generator $\mathcal{G}(1^\lambda)$, consisting of a prime p that specify the finite field $\mathbb{F}_p$, two elements $a, b \in \mathbb{F}_p$ that specify an elliptic curve $E(\mathbb{F}_p)$ defined by the equation: $E : y^2 = x^3 + ax + b \pmod{p}$, a base point $g = (x_g, y_g)$ on $E(\mathbb{F}_p)$, and a prime q which is the order of g. We denote the cyclic group generated by g as $\mathbb{G}$, and it is assumed that the DDH assumption holds over $\mathbb{G}$. More specifically, our commitment scheme consists of the following algorithms:

- $\mathsf{Gen}(\mathsf{Param}, 1^\lambda)$: picks $x \leftarrow \mathbb{Z}_q$, sets $h := g^x$, and outputs $\mathsf{ck} := (\mathsf{Param}, h)$.
- $\mathsf{Com}_{\mathsf{ck}}(m; r)$: outputs $c := (g^r, g^m h^r)$.
- $\mathsf{Ver}_{\mathsf{ck}}(c; m; r)$: outputs accept if $c = (g^r, g^m h^r)$; otherwise, outputs reject.

It is obvious that the above commitment scheme is perfectly binding and computationally hiding under the DDH assumption, i.e. for any PPT adversary $\mathcal{A}$, we have that the advantage

$$\mathsf{Adv}_{\mathrm{hide}}(\mathcal{A}) := \left| \Pr\left[\begin{array}{l} \mathsf{Param} \leftarrow \mathcal{G}(1^\lambda); \mathsf{ck} \leftarrow \mathsf{Gen}(\mathsf{Param}, 1^\lambda); \\ (m_0, m_1) \leftarrow \mathcal{A}(\mathsf{Param}, \mathsf{ck}); b \leftarrow \{0,1\}; \\ r \leftarrow \mathbb{Z}_q : \mathcal{A}(\mathsf{Com}_{\mathsf{ck}}(m_b; r)) = b \end{array} \right] - 1/2 \right|$$

is negligible in λ. The commitment scheme is additively homomorphic. Namely,

$$\mathsf{Com}_{\mathsf{ck}}(m_1; r_1) \cdot \mathsf{Com}_{\mathsf{ck}}(m_2; r_2) = \mathsf{Com}_{\mathsf{ck}}(m_1 + m_2; r_1 + r_2).$$

3.2 Schwartz-Zippel (min-entropy variant)

We need a min-entropy variant of the Schwartz-Zippel lemma, to check the equality of two univariate polynomials f_1, f_2, i.e. test $f_1(x) - f_2(x) = 0$ for random $x \overset{D}{\leftarrow} \mathbb{Z}_q$. The probability that the test passes is at most $\frac{\max(d_1,d_2)}{2^\kappa}$ if $f_1 \neq f_2$, where d_i is the degree of f_i for $i \in \{1,2\}$. We state the following lemma without proof (a proof will be provided in the full version).

Lemma 1 (min-entropy Schwartz-Zippel) *Let $f(x)$ be a non-zero univariate polynomial of degree d over $\mathbb{Z}_q$. Let D be a probability distribution on $\mathbb{Z}_q$ such that $H_\infty(D) \geq \kappa$. The probability of $f(x) = 0$ for a randomly chosen $x \overset{D}{\leftarrow} \mathbb{Z}_q$ is at most $\frac{d}{2^\kappa}$.*

3.3 A Σ Protocol for Candidate Encoding Correctness

In order to present the Σ protocol with clarity, we outline some necessary excerpts of the description of our system that will be explained in detail in Section 3.5.

Let $N = n + 1$, where n is the number of voters. Each voter is given a ballot that consists of two equivalent parts that contain a list of m vote-codes corresponding to the list candidates $\{P_1, \ldots, P_m\}$. The voter will flip a coin to choose the part she is going to use for voting. At the **Setup** phase, each ballot is posted to the BB in committed form. Namely, it consists of two sets of commitments $E^{(a)}_{\ell,j}$ for $a \in \{0,1\}, \ell = 1, \ldots, n, j = 1, \ldots, m$, and each set commits to a permutation of the encoded candidates, where candidate P_j is encoded as N^{j-1}.

We emphasize that it is not necessary to prove that each set of the commitments commits to a permutation of the encoded candidates $\{N^0, \ldots, N^{m-1}\}$ in an 1-out-of-m election. This is due to two facts: (i) EA will open one of the two sets of commitments according to the corresponding voter's coin a_ℓ (the set that corresponds to the unused ballot part); therefore, a malicious EA will be caught with probability $1/2$ by each honest voter if any of the committed sets is not a permutation of the encoded candidates or is an inconsistent permutation of the encoded candidates w.r.t. the one on the voter's ballot. (ii) Even if we ensure that the set of the commitments commits to a permutation of the encoded candidates, it does not imply that the permutation is consistent to the one on the voter's ballot. In an 1-out-of-m election, only one of the commitments will be used for tally, and thus proving that the set of the commitments commits to an unknown permutation of the encoded candidates can only provide the guarantee that the tallied commitment commits to an encoded candidate. Note that this guarantee is important; otherwise, given that we perform homomorphic tallying, it may be feasible for a cheating EA to introduce a large deviation to the actual tally result via a single inconsistent ballot; for instance, EA may commit to $10000 \cdot N^{j-1}$ for some $j \in [m]$. Hence, we want the EA to show that each commitment commits to one of N^{j-1} for $j \in [m]$. [4] We can formalize the correctness of a single commitment problem as follows. Given commitments E, the prover wants to convince the verifier that he knows $r \in \mathbb{Z}_q$ such that $E = \mathsf{Com}_{\mathsf{ck}}(N^i; r)$ and $i \in [0, m-1]$. Let i, r be the prover's private input, and w.l.o.g. we assume m is a perfect power of 2. For general cases, say $2^{e-1} \leq m \leq 2^e$, we can show the conjunction $i \in [0, 2^e] \wedge (i + 2^e - m) \in [0, 2^e]$. Our Σ Protocol is described in Fig. 3.

Theorem 1 *Let $N > 0$ be a public integer. Given common input $E \in \mathbb{G} \times \mathbb{G}$, the protocol described in Fig. 3 is a Σ protocol for knowledge of $i \in \mathbb{N}, r \in \mathbb{Z}_q$ such that $E = \mathsf{Com}_{\mathsf{ck}}(N^i; r)$, $i \in [0, m-1]$ that is perfectly complete, statistically sound with soundness error $2^{-\kappa+1+\log\log m}$ when the verifier's challenge has min-entropy κ and computationally zero-knowledge with distinguishing advantage $\mathsf{Adv}_{\mathsf{zk}}(\mathcal{A}) \leq \log m \cdot \mathsf{Adv}_{\mathrm{hide}}(\mathcal{A})$ for any PPT adversary $\mathcal{A}$.*

[4]For efficiency, EA is only required to show the commitments that are used for tally commit to valid encoded candidates. On the other hand, since EA cannot predicate which commitments are going to be used for tally before the election, she has to prepare all the Σ protocols in the **Setup** phase; whereas she is only required to complete those Σ protocols for the commitment that will be tallied in the **Tally** phase.

$P(i, r)$**:**

Define b_j such that $i = \sum_{j=0}^{\log m - 1} b_j 2^j$. Pick

- $t_j, z_j, y_j, r_j, w_j, f_j \leftarrow \mathbb{Z}_q$ for $j \in [0, \log m - 1]$.

Compute the following commitments:

- For $j \in [0, \log m - 1]$,
  - $B_j = \mathsf{Com}_{\mathsf{ck}}(b_j; r_j)$; $T_j = \mathsf{Com}_{\mathsf{ck}}(t_j; z_j)$;
  - $Y_j = \mathsf{Com}_{\mathsf{ck}}((1 - b_j)t_j; y_j)$;
  - $W_j = \mathsf{Com}_{\mathsf{ck}}(w_j; f_j)$.

Define A_j, a_j, r'_j such that $A_j = B_j^{N^{2^j} - 1} \cdot \mathsf{Com}_{\mathsf{ck}}(1; 0) = \mathsf{Com}_{\mathsf{ck}}(a_j; r'_j)$, for $j \in [0, \log m - 1]$. Define $\{\beta_j, \gamma_j\}_{j=0}^{\log m}$ such that $\prod_{j=0}^{\log m - 1}(a_j X + w_j) = \sum_{j=0}^{\log m} \beta_j X^j$ and $\prod_{j=0}^{\log m - 1}(r'_j X + f_j) = \sum_{j=0}^{\log m} \gamma_j X^j$. (Note that for efficiency reasons, the prover needs to choose the $\{r_j\}_{j=0}^{\log m - 1}$ such that $\gamma_{\log m} = r$ in previous step.)

- For $j \in [0, \log m - 1]$, $D_j = \mathsf{Com}_{\mathsf{ck}}(\beta_j; \gamma_j)$.

$$\boxed{\begin{array}{l}\text{Return } \phi_1 = \{B_j, T_j, Y_j, W_j, D_j\}_{j=0}^{\log m - 1} \text{ and} \\ \\ \text{state}_\phi = \{t_j, z_j, y_j, r_j, b_j, w_j, f_j\}_{j=0}^{\log m - 1}.\end{array}}$$

$P \to V$**:** Send ϕ_1.

$V \to P$**:** $\boxed{\text{Send } \rho \leftarrow \mathbb{Z}_q.}$

$P(\text{state}_\phi)$**:** Compute the following answers:

- For $j \in [0, \log m - 1]$,
  - $t'_j = b_j \rho + t_j, z'_j = r_j \rho + z_j, y'_j = -y_j - r_j t'_j$;
  - $w'_j = a_j \rho + w_j, f'_j = r'_j \rho + f_j$;

$$\boxed{\text{Set } \phi_2 = \{t'_j, z'_j, y'_j, w'_j, f'_j\}_{j=0}^{\log m - 1}.}$$

$P \to V$**:** send ϕ_2

$V(E, \phi_1, \rho, \phi_2)$**:** Accept the proof (i.e. output accept) if and only if

- For $j \in [0, \log m - 1]$,
  - $B_j^\rho \cdot T_j = \mathsf{Com}_{\mathsf{ck}}(t'_j, z'_j)$,
  - $(\mathsf{Com}_{\mathsf{ck}}(1; 0)/B_j)^{t'_j} / Y_j = \mathsf{Com}_{\mathsf{ck}}(0; y'_j)$;
  - $A_j^\rho \cdot W_j = \mathsf{Com}_{\mathsf{ck}}(w'_j, f'_j)$;
- $E^{\rho^{\log m}} \prod_{j=0}^{\log m - 1} D_j^{\rho^j} = \mathsf{Com}_{\mathsf{ck}}(\prod_{j=0}^{\log m - 1} w'_j; \prod_{j=0}^{\log m - 1} f'_j)$;

Figure 3: The Σ Protocol for Ballot Correctness

Proof: It is straightforward to check that protocol in Fig. 3 achieves perfect completeness.

In terms of statistical soundness, the protocol verifies two facts. Namely, (i) $\{B_j\}_{j \in [0, \log m - 1]}$ commits to either 0 or 1, and (ii) E commits to $N^{\sum_{j=0}^{\log m - 1} b_j 2^j} = N^i$, where b_j is the opening of B_j. To check the first fact, for each committed b_j the protocol builds the degree 1 polynomial

$$g_1(X) = (1 - b_j)(b_j X + t) + c_0 = (1 - b_j) b_j X + c'_0$$

for some c_0 and c'_0. By min-entropy Schwartz-Zippel Lemma 1, if $H_\infty(\rho) \geq \kappa$ and $g_1(\rho) = 0$, the probability $\Pr[(1-b_j)b_j \neq 0] \leq 2^{-\kappa}$. Hence, with at least $1-2^{-\kappa}$ probability $(1-b_j)b_j = 0$, which implies $b_j \in \{0,1\}$. To check the second fact, the protocol first computes $A_j = B_j^{N^{2^j}-1} \cdot \mathsf{Com}_{\mathsf{ck}}(1;0)$ homomorphically. Let a_j be the opening of A_j. It is easy to see that $a_j = N^{2^j}$ if $b_j = 1$, $a_j = 1$ if $b_j = 0$, thus it holds that $a_j = b_j N^{2^j} + 1 - b_j = N^{b_j 2^j}$. So that the protocol just needs to verify that E commits to the product of a_j's. The verifier checks equality between two degree $\log m$ polynomials

$$g_2(X) = \prod_{j=0}^{\log m-1}(a_j X + w_j) = \sum_{j=0}^{\log m} \beta_j X^j \text{ and } g_2'(X) = uX^{\log m} + \sum_{j=0}^{\log m-1} \beta_j^* X^j$$

where u is the opening of E and β_j^* which is the opening of D_j and are provided by the (potentially malicious) prover. By min-entropy Schwartz-Zippel lemma, if $H_\infty(\rho) \geq \kappa$ and $g_2(\rho) = g_2'(\rho)$, the probability $\Pr[u = \beta_{\log m}] \geq 1 - \frac{\log m}{2^\kappa}$. Hence, we have $u = N^{\sum_{j=0}^{\log m-1} b_j 2^j}$ with at least $1 - \frac{\log m}{2^\kappa}$ probability conditioned on the fact (i). Given that all $b_0, \ldots, b_{\log m-1}$ need to be shown in $\{0,1\}$ the entire proof is statistically sound with probability $(1-2^{-\kappa})^{\log m}(1 - \frac{\log m}{2^\kappa}) \geq 1 - \log m \cdot 2^{-\kappa+1}$.

Our protocol satisfies special soundness, i.e. there exists an extractor that can extract $i \in \mathbb{N}, r \in \mathbb{Z}_q$ if the prover is able to complete the protocol twice with the same ϕ_1 but two distinct challenges (we omit the construction of the extractor).

To show special honest verifier zero-knowledge property, we now construct a simulator that on input $\hat{\rho} \in \mathbb{Z}_q$ can output a transcript that is indistinguishable from the real one. The simulator randomly picks $b_0, \ldots, b_{\log m-1} \leftarrow \{0,1\}$ and generates $\left\{t_j, z_j, y_j, r_j, B_j, T_j, Y_j, t_j', z_j', y_j', w_j, f_j, W_j, w_j', f_j'\right\}_{j=0}^{\log m-1}$ according to the protocol description. It then generates $\{D_j\}_{j=1}^{\log m-1}$ according to the protocol and set

$$D_0 = \mathsf{Com}_{\mathsf{ck}}(\prod_{j=0}^{\log m-1} w_j'; \prod_{j=0}^{\log m-1} f_j')/(E^{\hat{\rho}^{\log m}} \prod_{j=1}^{\log m-1} D_j^{\hat{\rho}^j}) \ .$$

Subsequentely, the simulator sets $\hat{\phi}_1 = \{B_j, T_j, Y_j, W_j, D_j\}_{j=0}^{\log m-1}$ and $\hat{\phi}_2 = \left\{t_j', z_j', y_j', w_j', f_j'\right\}_{j=0}^{\log m-1}$, and it outputs $(\hat{\phi}_1, \hat{\rho}, \hat{\phi}_2)$. First of all, it is obvious that all the verification equations hold. Secondly, the distribution of all the variables in $\hat{\phi}_2$ are uniformly random, which is identical to that of a real transcript. Moreover, if the adversary can distinguish the simulated $\hat{\phi}_1$ from that of a real transcript, she must be able to distinguish at least one of the fake $\{B_j\}_{j=0}^{\log m-1}$. By hybrid argument, we have for any PPT adversary $\mathcal{A}$, the advantage to distinguish the simulated proof is $\mathsf{Adv}_{\mathrm{zk}}(\mathcal{A}) \leq \log m \cdot \mathsf{Adv}_{\mathrm{hide}}(\mathcal{A})$. □

3.4 Producing the Verifier's Challenges

The main difficulty in our setting is that we would like to extract the challenge of the Σ protocol from the voters' coins $\mathbf{a} = \langle a_1, \ldots, a_n \rangle \in \{0,1\}^n$ using a deterministic algorithm. Recall that some of the voters might be malicious and colluding with the EA, so the entropy of the voters' coins is only contributed by the honest voters while the malicious voters' coins can depend on the honest ones. Note that the voters' coins should be ordered by their serial numbers, rather than their submission order. This is because in the latter case, the adversary can schedule the **Cast** protocols of all voters at will and as a result reduce the min-entropy of $\mathbf{a}$ to be at most $\log \theta$ where θ is the number of honest voters. Such level of entropy is insufficient to provide a sufficiently small verifiability error (that ideally drops exponentially with θ). For all the uncast ballots, we set their corresponding coins to 0 by default; therefore, $\mathbf{a}$ is always an n-bit source, regardless of the number of voters that complete the **Cast** protocol. We observe that the voters' coins $\mathbf{a}$ is a weaker source compared to a *non-oblivious bit-fixing source* [32], as the adversary is able to choose which bit(s) to fix during the coin flipping (source generation) process. On the other hand, if we restrict the adversary $\mathcal{A}$ in our verifiability game from being capable of scheduling **Cast** protocols freely and all voters have to submit their votes sequentially according to a pre-determined order in the ballot casting stage, the source $\mathbf{a}$ can be viewed as an *adaptive bit-fixing source* [38]; in such case, we can employ the deterministic extractor construction framework from [32] which applies a deterministic low influence function on segments of the source. The majority function is proven to be an optimal low influence function thus in this way we obtain a deterministic extractor that generates the challenge. However, this adversarial setting is not realistic in practice as ballot casting might be scheduled adversarially. Nevertheless, we emphasise that even using a non-oblivious bit-fixing source, Kamp and Zuckerman showed that at most n/ℓ bits can be extracted when ℓ out of n bits are fixed [32]. This result implies that if a deterministic extractor is used to generate $\Theta(\lambda)$ random bits, then this will restrict the percentage of corrupted voters to be below $\Theta(\frac{1}{\lambda})$ which might also be not a realistic expectation in practice. An alternative approach may use a condenser as opposed to an extractor. Randomized condensers with a small/constant seed space have been put forth see e.g. [2, 44]; using such a tool one may iterate over all possible seeds and thus be assured that one of the seeds will allow the condenser to produce a sufficiently random challenge. For instance, Barak *et al.* [2] proposed a basic 2-bit seed condenser $\mathsf{con} : \{0,1\}^n \to (\{0,1\}^{n/3})^4$ such that for every δ-source X with $0 < \delta < 0.9$, at least one of the 4 output blocks of $\mathsf{con}(X)$ is a $(\delta + \Omega(\delta^2))$-source. Based on the composing lemma (Lemma 5.5 [2]), we can iteratively apply the condenser to achieve any desired constant rate. Given a c-coin condenser $\mathrm{Con} : \{0,1\}^n \mapsto (\{0,1\}^\ell)^c$, in order to produce a good challenge, by definition, it should hold that $c \cdot \ell > n$, which means that the condenser will produce c blocks, one of which is guaranteed to be sufficiently random. However as we observe below, we can utilize ZK amplification to obtain essentially the same result as with a c-coin condenser while sacrificing very little entropy from the weak source. We explain our technique next.

Let $\{0,1\}^{\ell_\Sigma}$ be the challenge space, where $\ell_\Sigma = \lfloor \log q \rfloor$ and q is the order of the underlying group used in the Σ protocol. Assume $n/k \le \ell_\Sigma$ for some $k \in \mathbb{Z}^+$. We evenly partition the voters' coins $\mathbf{a}$ into k blocks, denoted by $\mathbf{a}_1, \dots, \mathbf{a}_k$. For each block $\mathbf{a}_i$, the EA should prove the correctness of the ballots using a separate Σ protocol with $\mathbf{a}_i$ as its challenge. The verifier only accepts the EA's proof if *all* the Σ protocols are valid. The theorem below shows that the soundness error of this k-times repeated Σ protocol drops exponentially with $\theta - k(\log\log m + 1)$.

Theorem 2 *Denote $\mathbf{a} = (\mathbf{a}_1, \dots, \mathbf{a}_k)$, and suppose $H_\infty(\mathbf{a}) = \theta$. For all adversarial prover $\mathcal{A}$, we have that*

$$\epsilon(m,n,k,\theta) = \Pr\left[\begin{array}{l} \mathsf{ck} \leftarrow \mathsf{Gen}(\mathsf{Param}, 1^\lambda); (E, x, r, \{\phi_{1,i}\}_{i=1}^k) \leftarrow \mathcal{A}(\mathsf{Param}, \mathsf{ck}); \\ \{\phi_{2,i}\}_{i=1}^k \leftarrow \mathcal{A}(\mathbf{a}_1, \dots, \mathbf{a}_k) : \mathsf{Ver}_{\mathsf{ck}}(E; x; r) = \mathsf{accept} \quad \wedge \\ x \notin \left\{N^0, \dots, N^{m-1}\right\} \ \wedge\ \forall i \in [k], V(E, \phi_{1,i}, \mathbf{a}_i, \phi_{2,i}) = \mathsf{accept} \end{array}\right]$$
$$\le 2^{k \log\log m - \theta + k}.$$

Proof: According to Theorem 1, for each challenge $\mathbf{a}_i$, the Σ protocol described in Fig. 3 is statistically sound with soundness error $\log m \cdot 2^{-H_\infty(\mathbf{a}_i)+1}$. Hence, for each challenge $\mathbf{a}_i, i \in [k]$, the probability $\mathsf{Ver}_{\mathsf{ck}}(E; x; r) = \mathsf{accept} \ \wedge\ x \notin \left\{n^0, \dots, n^{m-1}\right\} \ \wedge\ V(E, \phi_{1,i}, \mathbf{a}_i, \phi_{2,i}) = \mathsf{accept}$ is at most $2^{\log\log m - H_\infty(\mathbf{a}_i)+1}$. Therefore, we have the overall soundness error

$$\begin{aligned} \epsilon(m,n,k,\theta) &= \Pr\left[\begin{array}{l} \mathsf{ck} \leftarrow \mathsf{Gen}(\mathsf{Param}, 1^\lambda); (E, x, r, \{\phi_{1,i}\}_{i=1}^k) \leftarrow \mathcal{A}(\mathsf{Param}, \mathsf{ck}); \\ \{\phi_{2,i}\}_{i=1}^k \leftarrow \mathcal{A}(\mathbf{a}_1, \dots, \mathbf{a}_k) : \mathsf{Ver}_{\mathsf{ck}}(E; x; r) = \mathsf{accept} \quad \wedge \\ x \notin \left\{n^0, \dots, n^{m-1}\right\} \ \wedge\ \forall i \in [k], V(E, \phi_{1,i}, \mathbf{a}_i, \phi_{2,i}) = \mathsf{accept} \end{array}\right] \\ &\le \prod_{i=1}^k 2^{\log\log m - H_\infty(\mathbf{a}_i)+1} = 2^{k\log\log m - \sum_{i=1}^k H_\infty(\mathbf{a}_i) + k} = 2^{k \log\log m - H_\infty(\mathbf{a}) + k}. \end{aligned}$$

□

3.5 Description of our e-voting system

The description of our e-voting system follows the syntax in Section 2.2. For simplicity, we present our system for *1-out-of-m elections*, i.e. $\mathcal{U} = \{\{P_1\}, \dots, \{P_m\}\}$. The commitment scheme and the

Σ-protocol that are applied in our system, are the ones presented at length in sections 3.1 and 3.3 respectively.

Setup$(1^\lambda, \mathcal{P} = \{P_1, \ldots, P_m\}, \mathcal{V} = \{V_1, \ldots, V_n\}, \mathcal{U} = \{\{P_1\}, \ldots, \{P_m\}\})$**.** Let $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Ver})$ be the PPT algorithms that constitute the perfectly binding, computationally hiding and additively homomorphic commitment scheme presented in Section 3.1. The EA runs $\mathsf{Gen}(\mathsf{Param}, 1^\lambda)$ to generate the commitment key ck. Then, for $\ell \in [n]$, EA executes the following steps:

(i). It selects a unique label for the ℓ-th double ballot denoted by tag_ℓ.

(ii). It selects random permutations $\pi_\ell^{(0)}, \pi_\ell^{(1)}$ over $[m]$. The use of $\pi_\ell^{(0)}$ (reps. $\pi_\ell^{(1)}$) is to shuffle the order that the (vote-code, candidate) pairs in the part $s_\ell^{(0)}$ (resp. $s_\ell^{(1)}$) of the *double ballot* s_ℓ will be posted on the BB (in committed form), in order to support privacy.

(iii). For $j \in [m]$, it selects unique vote-codes $C_{\ell,j}^{(0)}, C_{\ell,j}^{(1)} \leftarrow \mathbb{Z}_q$, where q is the size of the group of the commitment scheme[5]. The vote-code $C_{\ell,j}^{(0)}$ (resp. $C_{\ell,j}^{(1)}$) is the one that will be associated with candidate P_j in part $s_\ell^{(0)}$ (resp. $s_\ell^{(1)}$) of s_ℓ.

(iv). For $a \in \{0, 1\}$, it prepares the ballot part $s_\ell^{(a)} = \left\{\left(P_j, C_{\ell,j}^{(a)}\right)\right\}_{j\in[m]}$ and generates the ballot

$$\boxed{s_\ell = \left(\mathrm{tag}_\ell, s_\ell^{(0)}, s_\ell^{(1)}\right)}\,.$$

(v). For $j \in [m]$, it computes $j' = \pi_\ell^{(a)}(j)$ and

- For $a \in \{0, 1\}$ (where a indicates the part $s_\ell^{(a)}$ of s_ℓ), it chooses randomness $t_{\ell,j'}^{(a)} \leftarrow \mathbb{Z}_q$ and computes the *vote-code commitment* for $C_{\ell,j'}^{(a)}$:

$$\boxed{U_{\ell,j'}^{(a)} = \mathsf{Com}_{\mathsf{ck}}\big(C_{\ell,j'}^{(a)}; t_{\ell,j'}^{(a)}\big)}\,.$$

- For $a \in \{0, 1\}$, it chooses randomness $r_{\ell,j'}^{(a)} \leftarrow \mathbb{Z}_q$ and computes the *encoded candidate commitment* for $P_{j'}$:

$$\boxed{E_{\ell,j'}^{(a)} = \mathsf{Com}_{\mathsf{ck}}\big((n+1)^{j'-1}; r_{\ell,j'}^{(a)}\big)}\,,$$

  where $(n + 1)^{j'-1}$ is the *encoding* of candidate $P_{j'}$. This encoding is selected to ensure the correctness of our system, as we show in Theorem 3.

- For $a \in \{0, 1\}$, EA prepares *pre-audit data* $\phi_{1,\ell,j'}^{(a)}$ to be used for verifying that $E_{\ell,j'}^{(a)}$ is a commitment to a valid encoding from the set $\{(n + 1)^0, \ldots, (n + 1)^{m-1}\}$ at the verification phase. In addition, it maintains *prover state* $\mathrm{state}_{\phi,\ell,j'}^{(a)}$. Both $\phi_{1,\ell,j'}^{(a)}$ and $\mathrm{state}_{\phi,\ell,j'}^{(a)}$ are described in the Σ-protocol shown in Figure 3 (first move) of Section 3.3.

(vi). $\mathrm{Pub}_\ell = \left(\mathrm{tag}_\ell, \left\{\big(U_{\ell,j'}^{(a)}, E_{\ell,j'}^{(a)}, \phi_{1,\ell,j'}^{(a)}\big)\right\}_{j\in[m]}^{a\in\{0,1\}}\right)$ is the public information w.r.t. s_ℓ. It is indexed by tag_ℓ and contains the ballot information for both parts in committed form, as well as the respective pre-audit data. The information that refers to the (vote-code, candidate) pair $(C_{\ell,j'}^{(a)}, P_{j'})$ is tabulated in the j-th location of the part that is associated with $s_\ell^{(a)}$.

[5]For simplicity in presentation, we commit to the vote-codes using the homomorphic commitment scheme of Section 3.1. We stress that since no arithmetic operations are executed in the vote-code commitments, we could use more efficient commitment schemes and in this case vote-codes may be drawn from a domain that is smaller than $\mathbb{Z}_q$ resulting in a more “user-friendly” interface.

The public information that EA generates is

$$\boxed{\mathrm{Pub} = (\mathsf{ck}, \mathcal{P}, \mathcal{U}, \{\mathrm{Pub}_\ell\}_{\ell\in[n]})} \,.$$

The secret key of EA is

$$\boxed{msk = \{\mathrm{Pub}_\ell, s_\ell, msk_\ell, \mathrm{state}_{\phi,\ell}\}_{\ell\in[n]}} \,,$$

where we denote $msk_\ell = \left\{(C^{(a)}_{\ell,j}, t^{(a)}_{\ell,j}, \pi^{(a)}_\ell(j) = j', r^{(a)}_{\ell,j})\right\}^{a\in\{0,1\}}_{j\in[m]}$ and $\mathrm{state}_{\phi,\ell} = \left\{\mathrm{state}^{(a)}_{\phi,\ell,j'}\right\}^{a\in\{0,1\}}_{j\in[m]}$.

The Cast protocol. On input $(\mathrm{Pub}, s_\ell, \mathcal{U}_\ell)$, voter V_ℓ flips a coin $a_\ell \leftarrow \{0,1\}$ and picks part $s^{(a_\ell)}_\ell$ to vote and part $s^{(a_\ell)}_\ell$ for audit. Let P_{j_ℓ} be the candidate that V_ℓ is going to vote for, i.e., $\mathcal{U}_\ell = \{P_{j_\ell}\}$. Then, V_ℓ selects to submit $C^{(a_\ell)}_{\ell,j_\ell}$, which is the vote-code that corresponds to P_{j_ℓ} in part $s^{(a_\ell)}_\ell$. Next, V_ℓ casts the vote $\psi_\ell = \left(\mathrm{tag}_\ell, a_\ell, C^{(a_\ell)}_{\ell,j_\ell}\right)$. The EA receives the vote and updates its state st by appending ψ_ℓ. The receipt α_ℓ of V_ℓ is the vote ψ_ℓ and the part $s^{(1-a_\ell)}_\ell$ used for audit.

The Tally(Pub) **protocol.** Let $\tilde{\mathcal{V}}$ be the set of the voters that have voted successfully.

- For each $V_\ell \in \tilde{\mathcal{V}}$, the EA uses $(\mathrm{tag}_\ell, a_\ell)$ from ψ_ℓ to recover the respective audit information $s^{(1-a_\ell)}_\ell$ from s_ℓ. Then, it sends to BB the list $\left\{(\psi_\ell, s^{(1-a_\ell)}_\ell)\right\}_{V_\ell\in\tilde{\mathcal{V}}}$. It also opens all the vote-code commitments, $\left\{U^{(a)}_{\ell,j}\right\}^{a\in\{0,1\}}_{\ell\in[n],j\in[m]}$, by sending the list of pairs $\left\{(C^{(a)}_{\ell,j}, t^{(a)}_{\ell,j})\right\}^{a\in\{0,1\}}_{\ell\in[n],j\in[m]}$ to the BB.
- The EA, for every ψ_ℓ corresponding to a $V_\ell \in \tilde{\mathcal{V}}$:
  - (i). locates the decommitted vote-code C_ℓ that matches the cast vote-code $C^{(a_\ell)}_{\ell,j_\ell}$. Then, it marks the vote-code C_ℓ as 'voted' and adds the corresponding commitment $E^{(a_\ell)}_{\ell,j'_\ell}$ into the set $\mathbf{E}_{\mathrm{tally}}$ (initially empty). Recall that $j'_\ell = \pi^{(a_\ell)}_\ell(j_\ell)$.
  - (ii). adds all the commitments $\{E^{(1-a_\ell)}_{\ell,j}\}_{j\in[m]}$ that correspond to the vote-codes in $s^{(1-a_\ell)}_\ell$ into the set $\mathbf{E}_{\mathrm{open}}$ (initially empty).

  When finalised, $\mathbf{E}_{\mathrm{tally}}$ includes the collection of votes that will be counted (homomorphically) and $\mathbf{E}_{\mathrm{open}}$ includes the information that will be used for verifying ballot correctness. After this happens, EA posts to the BB the list of marked vote-codes along with $\mathbf{E}_{\mathrm{tally}}$ and $\mathbf{E}_{\mathrm{open}}$.
- The EA produces and posts to the BB all the *verifier's challenges* $\{\rho_E\}_{E\in\mathbf{E}_{\mathrm{tally}}}$ of the Σ-protocols for the validity of the commitments in $\mathbf{E}_{\mathrm{tally}}$, as determined in Figure 3 (second move). The extraction of the challenges is done via the randomness contributed by the voters' coin-flips. The extraction method that is used is described in Section 3.4.
- The EA prepares and posts to the BB all the *post-audit data* $\{\phi_{2,E}\}_{E\in\mathbf{E}_{\mathrm{tally}}}$ of the Σ-protocols for verifying the validity of the commitments in $\mathbf{E}_{\mathrm{tally}}$, as determined in Figure 3 (third move). Thus, for each commitment in $\mathbf{E}_{\mathrm{tally}}$ there is a triple of pre-audit data, challenge and post-audit data that form a *complete* Σ *proof* of a valid commitment to some encoded candidate.
- EA performs homomorphic tally by computing $E_{\mathrm{sum}} = \prod_{E\in\mathbf{E}_{\mathrm{tally}}} E$ and preparing (T, R) as the opening of E_{sum}. The additive homomorphic property implies that T is the election result encoded in the number system with base $N = n+1$ and it is committed under randomness R, which is the sum of all the randomness used for the commitments in $\mathbf{E}_{\mathrm{tally}}$. Next, EA opens all the commitments in $\mathbf{E}_{\mathrm{open}}$. Let Open be the set of these openings. Finally, it sends Open, E_{sum} and (T, R) to the BB.

- In the end of the process, BB contains the list of the marked vote-codes, as well as

$$\mathrm{Pub}, \left\{(C_{\ell,j}^{(a)}, t_{\ell,j}^{(a)})\right\}_{\ell\in[n],j\in[m]}^{a\in\{0,1\}}, \big(\mathbf{E}_{\mathrm{tally}}, E_{\mathrm{sum}}, (T,R)\big),$$

$$(\mathrm{Open}, \mathbf{E}_{\mathrm{open}}), \{\rho_E\}_{E\in\mathbf{E}_{\mathrm{tally}}}, \{\phi_{2,E}\}_{E\in\mathbf{E}_{\mathrm{tally}}}.$$

Result(τ). The election result R_τ is derived by the following decoding algorithm:

```
Set $X \leftarrow T$;
For $j = 1, \ldots, m$:
• $x_j \leftarrow X \bmod (n+1)$;
• $X \leftarrow (X - x_j)/(n+1)$;
Return $\langle x_1, \ldots, x_m \rangle$;
```

The correctness of the algorithm (and our system) is shown in Theorem 3.

Verify(τ, α). Initially, α is parsed as $\big(\mathrm{tag}, a, C, s^{(1-a)}\big)$. The algorithm returns 1 only if the following checks are valid:

(i). All committed information in τ is associated with n ballots indexed under different tags and no two vote-codes under the same tag are marked as 'voted'.

(ii). Let $\hat{C}$ be a vote-code that appears in part $\hat{s}^{(\hat{a})}$ of some ballot and has been marked as 'voted' . Then, only the committed information for the other part $\hat{s}^{(1-\hat{a})}$ of this ballot has been opened.

(iii). All the complete Σ proofs that are associated with commitments in $\mathbf{E}_{\mathrm{tally}}$ are valid.

(iv). $E_{\mathrm{sum}} = \prod_{E\in\mathbf{E}_{\mathrm{tally}}} E$.

(v). All the openings of the commitments are valid.

(vi). tag equals some tag_ℓ in τ for some $\ell \in [n]$ and it holds that $a = a_\ell$.

(vii). The vote-code that is marked as 'voted' and is associated to tag_ℓ is C where ℓ is as in item (vi).

(viii). The correspondence of candidate encodings to vote-codes revealed in the opening of the commitments $\{(U_{\ell,j}^{(1-a_\ell)}, E_{\ell,j}^{(1-a_\ell)})\}_{j\in[m]}$ where ℓ is as in item (vi), is equal to the one defined in $s^{(1-a)}$.

3.6 Correctness of our e-voting system

We prove the correctness of our system in the following theorem. In the remaining of the paper, we assume that $n \cdot (n+1)^{m-1} < q$.

Theorem 3 *Let q be the size of the group for the commitment scheme described in Section 3.1 and assume that $n\cdot(n+1)^{m-1} < q$. Then, the e-voting system described in Section 3.5 has perfect correctness.*

Proof: It is straightforward that in a honest execution where the information is consistently tabulated, all verifications are successful, by the correctness of the commitment scheme and the completeness of the Σ-protocol that are used. Thus, it suffices to show the correctness of the **Result**$(\cdot)$ algorithm.

We denote by P_{j_ℓ} the candidate that the voter V_ℓ has selected, i.e. $\mathcal{U}_\ell = \{P_{j_\ell}\}$. The encoding of P_{j_ℓ} is $(n+1)^{j_\ell-1}$, therefore we have that

$$E_{\mathrm{sum}} = \prod_{\ell\in[n]} \mathsf{Com}_{\mathsf{ck}}\big((n+1)^{j_\ell-1}; r_{\ell,j_\ell}^{(a_\ell)}\big) = \mathsf{Com}_{\mathsf{ck}}(T;R)\ .$$

Due to the binding and homomorphic properties, the latter implies that if the candidates $P_1, \ldots, P_m$ have been voted $t_1, \ldots, t_m$ times respectively, then E_{sum} is opened to $T = \sum_{j=1}^{m} t_j \cdot (n+1)^{j-1} \bmod q$. We observe that $T \le n \cdot (n+1)^{m-1} < q$ (all n voters vote for candidate P_m). Therefore, $T \bmod q = T$, i.e. E_{sum} is opened to the actual result, $\langle t_1, \ldots, t_m \rangle$. Moreover, since $0 \le t_1, \ldots, t_m \le n$, we have that $t_i = t_i \bmod (n+1)$, for all i. By induction, we will show that the output $\langle x_1, \ldots, x_m \rangle$ of the **Result** algorithm equals to $\langle t_1, \ldots, t_m \rangle$, which completes the proof.

- For $j = 1$, we have that

$$\begin{aligned} x_1 = T \bmod (n+1) &= \sum_{i=1}^{m} t_i \cdot (n+1)^{i-1} \bmod (n+1) = \\ &= t_1 \bmod (n+1) + (n+1) \cdot \sum_{i=2}^{m} t_i \cdot (n+1)^{i-2} \bmod (n+1) = t_1. \end{aligned}$$

- For $2 \le j \le m$, if $x_i = t_i$ for every $i < j$, then, by the description of the decoding algorithm

$$\begin{aligned} x_j &= \frac{T - \sum_{1 \le i < j} x_i \cdot (n+1)^{i-1}}{(n+1)^{j-1}} \bmod (n+1) = \\ &= \frac{\sum_{i=1}^{m} t_i \cdot (n+1)^{i-1} - \sum_{1 \le i < j} t_i \cdot (n+1)^{i-1}}{(n+1)^{j-1}} \bmod (n+1) = \\ &= \frac{\sum_{j \le i \le m} t_i \cdot (n+1)^{i-1}}{(n+1)^{j-1}} \bmod (n+1) = \\ &= t_j \bmod (n+1) + \sum_{0 < k \le m-j} t_{j+k} \cdot (n+1)^k \bmod (n+1) = t_j. \end{aligned}$$

□

3.7 Example of our e-Voting System

For the better understanding of our e-voting system, we provide a toy example of a referendum where $P_1 = $ YES, $P_2 = $ NO are the candidates and $\mathcal{V}$ consists of three voters V_1, V_2, V_3. Our goal is to familiarize the reader with the functionality of our system so, for simplicity, we deviate from the description in Section 3.5 by not including Σ-protocol proofs.

EA generates the vote-codes for the ballots s_1,s_2 and s_3 of V_1, V_2 and V_3 as

$$\begin{gathered} (C_{1,1}^{(0)} = 27935, C_{1,2}^{(0)} = 75218, C_{1,1}^{(1)} = 84439, C_{1,2}^{(1)} = 77396), \\ (C_{2,1}^{(0)} = 58729, C_{2,2}^{(0)} = 45343, C_{2,1}^{(1)} = 14582, C_{2,2}^{(1)} = 93484), \\ (C_{3,1}^{(0)} = 52658, C_{3,2}^{(0)} = 65864, C_{3,1}^{(1)} = 84373, C_{3,2}^{(1)} = 49251) \end{gathered}$$

respectively. The double ballots s_1, s_2, s_3 are labelled by the tags $101, 102, 103$ respectively and are formed as follows:

101	
27935	YES
75218	NO
84439	YES
77396	NO

102	
58729	YES
45343	NO
14582	YES
93484	NO

103	
52658	YES
65864	NO
84373	YES
49251	NO

EA prepares the commitments to each vote-code and the encoding of the candidate that they correspond. The commitment for YES (resp. NO) is a commitment to $(3+1)^0 = 1$ (resp. $(3+1)^1 = 4$).

Next, it chooses whether the commitments of the vote-code and candidate pairs are going to be ordered in the BB as they are in the ballot part, or swapped. For example, assume that for the ballot s_1, EA chooses to leave the order in ballot part (0) intact and to swap the pairs in ballot part (1). Then, the information posted in the BB for s_1 would have the following form:

101	
$\mathsf{Com}_{\mathsf{ck}}(27935; t_{1,1}^{(0)})$	$\mathsf{Com}_{\mathsf{ck}}(1; r_{1,1}^{(0)})$
$\mathsf{Com}_{\mathsf{ck}}(75218; t_{1,2}^{(0)})$	$\mathsf{Com}_{\mathsf{ck}}(4; r_{1,2}^{(0)})$
$\mathsf{Com}_{\mathsf{ck}}(77396; t_{1,2}^{(1)})$	$\mathsf{Com}_{\mathsf{ck}}(4; r_{1,2}^{(1)})$
$\mathsf{Com}_{\mathsf{ck}}(84439; t_{1,1}^{(1)})$	$\mathsf{Com}_{\mathsf{ck}}(1; r_{1,1}^{(1)})$

Suppose that V_1 votes for NO using ballot part (1), V_2 votes for YES using ballot part (1) and V_3 votes for YES using ballot part (0). Then, the votes cast by V_1, V_2 and V_3 are $(101, 1, 77396)$, $(102, 1, 14582)$ and $(103, 0, 52568)$ respectively. The receipts that the voters receive are

(101,1,77396)	
27935	YES
75218	NO

$(102, 1, 14582)$	
58729	YES
45343	NO

$(103, 0, 52568)$	
84373	YES
49251	NO

The coins that V_1, V_2 and V_3 have flipped, are $a_1 = 1, a_2 = 1$ and $a_3 = 0$ respectively. Hence, we get internal randomness, $(1, 1, 0)$, of 3 bits (which would be the "weak source" of randomness used for the extraction of the challenge of the Σ protocols). After the voting ends, EA opens the vote-code commitments, marks the cast vote-codes $77396, 14582$ and 52658 and includes the corresponding encoded candidate commitments $\mathsf{Com}_{\mathsf{ck}}(4; r_{1,2}^{(1)})$, $\mathsf{Com}_{\mathsf{ck}}(1; r_{2,1}^{(1)})$ and $\mathsf{Com}_{\mathsf{ck}}(1; r_{3,1}^{(0)})$ in the tally set. Next, EA performs homomorphic tally, by computing the product of the above encoded candidate commitments as

$$E_{\text{sum}} = \mathsf{Com}_{\mathsf{ck}}(4; r_{1,2}^{(1)}) \cdot \mathsf{Com}_{\mathsf{ck}}(1; r_{2,1}^{(1)}) \cdot \mathsf{Com}_{\mathsf{ck}}(1; r_{3,1}^{(0)}) = \mathsf{Com}_{\mathsf{ck}}(6; r_{1,2}^{(1)} + r_{2,1}^{(1)} + r_{3,1}^{(0)}).$$

Then, EA publishes E_{sum}, along with the opening of E_{sum} at value $(6; r_{1,2}^{(1)} + r_{2,1}^{(1)} + r_{3,1}^{(0)})$. The result is derived by computing $x_1 = 6 \bmod 4 = 2$ and $x_2 = ((6 - x_1)/4) \bmod 4 = 1$, which is interpreted as two votes for YES and one for NO.

In the verification phase, the EA opens the commitments in the ballot parts that the voters selected for auditing. For example, V_1 would check the consistency of her receipt with audit information in the BB, as illustrated below

101			
27935	YES	$(1, r_{1,1}^{(0)})$	$\mathsf{Com}_{\mathsf{ck}}(1; r_{1,1}^{(0)})$
75218	NO	$(4, r_{1,2}^{(0)})$	$\mathsf{Com}_{\mathsf{ck}}(4; r_{1,2}^{(0)})$
77396	VOTED		$\mathsf{Com}_{\mathsf{ck}}(4; r_{1,2}^{(1)})$
84439			$\mathsf{Com}_{\mathsf{ck}}(1; r_{1,1}^{(1)})$

Encodings	
YES	1
NO	4

Observe that, as we will prove, the cut-and-choose verification that V_1 performs, does not reveal her vote even to a party that obtains her receipt. This is because the cast vote-code alone does not leak any information about the associated candidate, while the entirely opened auditing part only serves as a check that the correspondence of the vote-codes and candidates in this part has not been tampered with. Therefore, V_1 can delegate the task of verification to a third party, without compromising her privacy.

4 Security of Our e-Voting System

In this section, we prove the security of our system in the definitional framework presented in Sections 2.4 and 2.3.

4.1 E2E Verifiability of Our e-Voting System

We prove that our e-voting system achieves E2E verifiability information theoretically in the standard model. We follow the notation in Figure 1 and the description in 3.5.

Theorem 4 *Let n be the number of all voters and m be the number of candidates. Let q be the size of the group for the commitment scheme described in Section 3.1. The e-voting system described in 3.5 achieves E2E verifiability information theoretically with error $(1/2)^d + \epsilon(m, n, \lceil n/\lfloor \log q \rfloor \rceil, \theta - 1))$, where θ is the number of honest successful voters, d is the tally deviation that the adversary wants to achieve and $\epsilon(m, n, \lceil n/\lfloor \log q \rfloor \rceil, \theta - 1)$ is the soundness error of the Σ protocol performed by the EA given in Theorem 2.*

Proof: Without loss of generality (w.l.o.g.), we assume that in any adversarial execution as described in the E2E verifiability game $G^{\mathcal{A},\mathcal{E},d,\theta}_{\mathrm{E2E-Ver}}(1^\lambda, m, n)$, exactly n ballots are tabulated on τ under n different tags and all vote-codes are marked as 'voted' correspond to different tags (if such deviations happen the transcript is immediately rejected). In the same spirit, we assume there is no double ballot that both parts have been opened and that all double ballots for honest voters in $\tilde{\mathcal{V}}$ are well-formed, otherwise they would not engage in the **Cast** protocol. Finally, we recall that the adversary cannot modify the history of the transcript since it does not have control over the BB. As a first step, we construct a vote extractor $\mathcal{E}$ for our system as follows:

Construction of the vote extractor. $\mathcal{E}$ has input τ and the set of receipts $\{\alpha_\ell\}_{V_\ell \in \tilde{\mathcal{V}}}$, where $\tilde{\mathcal{V}}$ is the set of the honest voters that voted successfully. Let $t \leq |\tilde{\mathcal{V}}|$ be the number of different tags that appear in $\{\alpha_\ell\}_{V_\ell \in \tilde{\mathcal{V}}}$[6]. If $\mathbf{Result}(\tau) = \perp$ (i.e., the transcript is not meaningful), then $\mathcal{E}$ outputs $\perp$. Otherwise, $\mathcal{E}$ (arbitrarily) arranges the voters in $\mathcal{V} \setminus \tilde{\mathcal{V}}$ and the tags not included in $\{\alpha_\ell\}_{V_\ell \in \tilde{\mathcal{V}}}$ as $\langle V^{\mathcal{E}}_\ell \rangle_{\ell \in [n - |\tilde{\mathcal{V}}|]}$ and $\langle \mathrm{tag}^{\mathcal{E}}_\ell \rangle_{\ell \in [n-t]}$ respectively. Next, for every $\ell \in [n - |\tilde{\mathcal{V}}|]$:

1. If there is no marked as 'voted' vote-code that is associated with $\mathrm{tag}^{\mathcal{E}}_\ell$, then $\mathcal{E}$ sets $\mathcal{U}^{\mathcal{E}}_\ell = \emptyset$ (encoded as the zero vector) which is interpreted as an abort for voter $V^{\mathcal{E}}_\ell$.

2. If there is a 'voted' vote-code $C^{(a)}_{\ell,j}$ that is associated with $\mathrm{tag}^{\mathcal{E}}_\ell$, then $\mathcal{E}$ brute-force opens the respective encoded candidate commitment $E^{(a)}_{\ell,j}$ to a value Open_ℓ (recall the commitment is perfectly binding). If Open_ℓ is a valid encoding (i.e. $\mathrm{Open}_\ell \in \{(n+1)^0, (n+1)^1, \ldots, (n+1)^{m-1}\}$) of a candidate $\mathcal{P}^{\mathcal{E}}_\ell$, then $\mathcal{E}$ sets $\mathcal{U}^{\mathcal{E}}_\ell = \{\mathcal{P}^{\mathcal{E}}_\ell\}$. Otherwise, it outputs $\perp$.

Finally, $\mathcal{E}$ outputs $\langle \mathcal{U}^{\mathcal{E}}_\ell \rangle_{V^{\mathcal{E}}_\ell \in \mathcal{V} \setminus \tilde{\mathcal{V}}}$. Note that if $t < |\tilde{\mathcal{V}}|$, then the remaining tags $\mathrm{tag}^{\mathcal{E}}_{n-|\tilde{\mathcal{V}}|+1}, \ldots, \mathrm{tag}^{\mathcal{E}}_{n-t}$ are ignored by $\mathcal{E}$.

Based on the above vote extractor, we will prove the E2E verifiability of our scheme. Assume an adversary $\mathcal{A}$ that wins the game $G^{\mathcal{A},\mathcal{E},d,\theta}_{\mathrm{E2E-Ver}}(1^\lambda, m, n)$. Namely, $\mathcal{A}$ breaks E2E verifiability by allowing at least θ honest successful voters and achieving tally deviation d. Since there is at least one honest voter that performs verification ($\theta > 0$), w.l.o.g. we assume that $\mathcal{A}$ always outputs meaningful transcripts.

Let F be the event that there exists a committed value in τ which is marked to be counted and invalid (i.e., it is in $\mathbf{E}_{\mathsf{tally}}$ but it is not a commitment to some candidate encoding). Since condition (i) of

[6]This implies that the ballot audit for all voters in $\tilde{\mathcal{V}}$ focuses on a list of t tabulated ballots on the BB. Thus, an adversary may inject $|\tilde{\mathcal{V}}| - t$ ballots for candidate selections of its choice that will be counted in the final tally as if they were honest.

$G^{\mathcal{A},\mathcal{E},d,\theta}_{\mathrm{E2E-Ver}}(1^\lambda, m, n)$ holds, we have that there are at least θ honest voters. However, the soundness error of the Σ- protocol is going to be affected by the fact that the invalid commitment is in a specific ballot part. The min entropy of all the coins given the fact that the adversary knows the coin of the invalid commitment in order to win, is at least the min entropy of all the coins minus 1 bit (i.e., the entropy of that bit). Therefore, by applying Theorem 2 for min entropy equal to $\theta - 1$, we have that each Σ protocol has soundness error $\epsilon(m, n, \lceil n/\lfloor \log q \rfloor \rceil, \theta - 1)$. Hence, the probability that a committed value is invalid while verification accepts is no more than $\epsilon(m, n, \lceil n/\lfloor \log q \rfloor \rceil, \theta - 1)$. Since there is at least one honest voter that verifies, we conclude that

$$\Pr[G^{\mathcal{A},\mathcal{E},d,\theta}_{\mathrm{E2E-Ver}}(1^\lambda, m, n) = 1 \wedge F] \leq \epsilon(m, n, \lceil n/\lfloor \log q \rfloor \rceil, \theta - 1). \tag{1}$$

Assume that F does not occur. Thus, all marked committed values in $\mathbf{E}_{\mathsf{tally}}$ correspond to a valid candidate encoding. This implies that (a) the maximum deviation per marked commitment that $\mathcal{A}$ may achieve is 1 (the vote is counted for a candidate other than the intended one) and (b) $\mathcal{E}$ does not output $\perp$ (it returns a vector $\langle \mathcal{U}^{\mathcal{E}}_\ell \rangle_{V^{\mathcal{E}}_\ell \in \mathcal{V} \setminus \tilde{\mathcal{V}}}$), so $\mathcal{A}$ wins because (i),(ii) and (iii-a) hold. The auditor can verify that E_{sum} is equal to the homomorphic commitment $\prod_{E \in \mathbf{E}_{\mathsf{tally}}} E$. Due to the perfect binding of the commitment scheme, the tally $f(\langle \mathcal{U}^{\mathcal{E}}_\ell \rangle_{V^{\mathcal{E}}_\ell \in \mathcal{V} \setminus \tilde{\mathcal{V}}})$ that $\mathcal{E}$ estimates as non-honest votes, is correctly included in the adversarial result that derives from the opening (T, R) of E_{sum}. Thus, the deviation from the intended result that $\mathcal{A}$ achieves, derives only by miscounting the honest votes. This may be achieved by $\mathcal{A}$ in two different possible ways:

1. **Modification attacks:** modify the committed information as compared with the one in an honest voter's ballot (e.g., alter the vote-code and candidate correspondence). The deviation achieved by this type of attack is at most 1.

2. **Clash attacks:** instruct r honest voters whose ballots are indexed under the same tag to vote so that the votes of any $r - 1$ out of these r voters are all different than some fixed $r - 1$ committed votes that are ignored by $\mathcal{E}$ (either cast by corrupted voters or initially injected in τ by $\mathcal{A}$). All r voters verify the correct counting of their votes by auditing the same information on the BB and hence miss the injected votes that produce the tally deviation. The deviation achieved by this type of attack is $r - 1$.

In the case where all ballot information is committed consistently on the BB without being deleted or replaced, the adversary can only perform a combination of these two attacks on the honest voters. Indeed, if all honestly cast votes are in one-to-one correspondence with the correct encoded candidate commitments, then the perfect binding property ensures that the opening of the homomorphic tally matches the intended result.

Let $\tilde{\mathcal{V}}_1, \ldots, \tilde{\mathcal{V}}_t$ be the partition of $\tilde{\mathcal{V}}$ s.t. each of these subsets consists of honest voters that their receipts (hence their ballots) are indexed under the same tag. These subsets are created adaptively, according to the strategy of $\mathcal{A}$, under the constraint that $|\tilde{\mathcal{V}}| \geq \theta$. Note that there are $|\tilde{\mathcal{V}}| - t$ ignored tags in vote extraction, while $\sum_{i \in [t]}(|\tilde{\mathcal{V}}_i| - 1) = |\tilde{\mathcal{V}}| - t$. This implies that the adversary can perform clash attacks in all these subsets, with maximum possible deviation. We will prove that given that F does not occur, the success probability of $\mathcal{A}$ is no more than $(1/2)^d$, whatever its strategy might be.

We observe that in order for $\mathcal{A}$ to win, all voters in $\mathcal{V}_i$ must have the same receipt, or else inconsistencies will cause verification to fail. To achieve this, $\mathcal{A}$ must instruct the voters from the same subset to vote so that they all cast the same vote-code (otherwise two marked vote-codes under the same tag should appear) and create the corresponding audit ballot part identically for each auditing voter. In detail, in order for $\mathcal{A}$ to win, the following must hold for each $\tilde{\mathcal{V}}_i$, $i \in [t]$:

1. There is a representative vote-code C_i that appears in part (a) of all the double ballots of the voters in $\tilde{\mathcal{V}}_i$. The voters must select this part to vote by casting C_i. Therefore, the coin-flippings of the auditing voters must be consistent, in the sense that they correspond to ballot parts that contain a consistent vote-code. There can be at most 2 consistent coin-flips (i.e., either all coins are flipped

to 0 or all coins are flipped to 1). Thus, the probability of consistent coin-flipping in $\tilde{\mathcal{V}}_i$ is at most $2/2^{|\tilde{\mathcal{V}}|_i} = (1/2)^{|\tilde{\mathcal{V}}|_i - 1}$. In addition, the ballot parts that will be used for auditing must contain the same information, up to a permutation of the vote-code and candidate pairs.

2. If $\mathcal{A}$ wants to achieve $|\tilde{\mathcal{V}}_i|$ deviation exploiting the voters in $\mathcal{V}_i$, then it must perform a modification attack in at least one voter V in $\tilde{\mathcal{V}}_i$. This is because if all voters' ballots are consistent to the corresponding committed information in τ, then by performing only a clash attack in $\tilde{\mathcal{V}}_i$, $\mathcal{A}$ can achieve deviation by at most $|\tilde{\mathcal{V}}_i| - 1$, as described above. However, the modification comes with a loss of 1/2 success probability, since $\mathcal{A}$ must also guess which is the part that V is going to use for voting. Indeed, if V chooses to audit the modified part of the ballot, then she will detect the attack. Therefore, all voters in $\mathcal{V}_i$ must perform a consistent coin-flip that agrees with the coin-flip of V. It is straightforward that in case of a single modification attack this event happens with $1/2 \cdot (1/2)^{|\tilde{\mathcal{V}}_i|-1} = (1/2)^{|\tilde{\mathcal{V}}_i|}$ probability. Moreover, in case $\tilde{\mathcal{V}}_i \geq 2$, performing two modification attacks does not lead to any improvement in terms of probability or maximum deviation.

We note that the above arguments hold trivially, if $\tilde{\mathcal{V}}_i$ is a singleton. Let $\mathbf{X}$ be the set of subsets from $\{\tilde{\mathcal{V}}_1, \ldots, \tilde{\mathcal{V}}_t\}$ that $\mathcal{A}$ performs clash attacks and $\mathbf{Y}$ the collection that $\mathcal{A}$ performs a modification attack on at least one voter in each of the subsets. According to the previous arguments, we have the following cases: (i) for each $\mathcal{V}_i \in \mathbf{X} \setminus \mathbf{Y}$ the maximum deviation is $|\tilde{\mathcal{V}}_i| - 1$, (ii) for each $\mathcal{V}_i \in \mathbf{Y} \setminus \mathbf{X}$ the maximum deviation is 1, (iii) for each $\mathcal{V}_i \in \mathbf{X} \cap \mathbf{Y}$ the maximum deviation is $|\tilde{\mathcal{V}}_i|$ and (iv) for each $\mathcal{V}_i \in \left\{\tilde{\mathcal{V}}_1, \ldots, \tilde{\mathcal{V}}_t\right\} \setminus (\mathbf{X} \cup \mathbf{Y})$ the maximum deviation is 0. For brevity, let $x = |\mathbf{X}|$ and $y = |\mathbf{Y}|$. Therefore, we have that the tally deviation from the intended result that $\mathcal{A}$ achieves is at most

$$\sum_{\mathcal{V}_i \in \mathbf{X} \setminus \mathbf{Y}} (|\tilde{\mathcal{V}}_i| - 1) + \sum_{\mathcal{V}_i \in \mathbf{Y} \setminus \mathbf{X}} 1 + \sum_{\mathcal{V}_i \in \mathbf{X} \cap \mathbf{Y}} |\tilde{\mathcal{V}}_i| = \sum_{\mathcal{V}_i \in \mathbf{X}} |\tilde{\mathcal{V}}_i| - x + y \leq |\tilde{\mathcal{V}}| - x + y.$$

We will now upper bound the success probability of $\mathcal{A}$. Since $\{\tilde{\mathcal{V}}_1, \ldots, \tilde{\mathcal{V}}_t\}$ is a partition of $\tilde{\mathcal{V}}$, we have that $\mathcal{A}$ must not be detected by all the voters in all of these subsets. So,

$$\begin{aligned}\Pr[G^{\mathcal{A},\mathcal{E},d,\theta}_{\mathrm{E2E-Ver}}(1^\lambda, m, n) = 1 | \neg F] &\leq \prod_{\mathcal{V}_i \in \mathbf{Y}} (1/2)^{|\tilde{\mathcal{V}}_i|} \cdot \prod_{\mathcal{V}_i \in \left\{\tilde{\mathcal{V}}_1, \ldots, \tilde{\mathcal{V}}_t\right\} \setminus \mathbf{Y}} (1/2)^{|\tilde{\mathcal{V}}_i| - 1} = \\ &= (1/2)^{\sum_{\mathcal{V}_i \in \mathbf{Y}} |\tilde{\mathcal{V}}_i| + \sum_{\mathcal{V}_i \in \{\tilde{\mathcal{V}}_1, \ldots, \tilde{\mathcal{V}}_t\} \setminus \mathbf{Y}} (|\tilde{\mathcal{V}}_i| - 1)} = \\ &= (1/2)^{|\tilde{\mathcal{V}}| - (t-y)} \leq (1/2)^{|\tilde{\mathcal{V}}| - x + y},\end{aligned}$$

because $x \leq t$. In order for $\mathcal{A}$ to win, it must hold that $|\tilde{\mathcal{V}}| - x + y \geq d$ (condition (iii-a) holds), therefore

$$\Pr[G^{\mathcal{A},\mathcal{E},d,\theta}_{\mathrm{E2E-Ver}}(1^\lambda, m, n) = 1 \wedge \neg F] \leq \Pr[G^{\mathcal{A},\mathcal{E},d,\theta}_{\mathrm{E2E-Ver}}(1^\lambda, m, n) = 1 | \neg F] \leq (1/2)^d. \tag{2}$$

By adding (1),(2) we conclude that

$$\Pr[G^{\mathcal{A},\mathcal{E},d,\theta}_{\mathrm{E2E-Ver}}(1^\lambda, m, n) = 1] \leq (1/2)^d + \epsilon(m, n, \lceil n / \lfloor \log q \rfloor \rceil, \theta - 1).$$

□

Remark. Note that if the number of honest voters satisfies the bound $\theta = \Omega(n \log\log m / \log q + \lambda)$, then the overall soundness error of the repeated Σ protocol will be sufficiently small. For instance, in an election where there are $n = 1000$ voters and $m = 40$ candidates we can use a group with at least 500 bit prime order q. Assuming a number of $\theta = 50$ honest voters (5% of total) we can divide the 1000 voter's coins into two challenges with 500 bits each (i.e. $k = 2$). With these parameters the above theorem will have a verifiability error that is at most $2^{-43} + (1/2)^d$ where d is the tally deviation. We remark that in this setting no deterministic extractor would be able to provide sufficient entropy and hence our ZK amplification technique is crucial.

4.2 Voter Privacy/Receipt Freeness of Our e-Voting System

In order to show our scheme satisfies privacy, we utilize complexity leveraging. Specifically, the system security parameter is configured such that breaking the hiding property of the underlying commitment scheme is much harder than guessing the challenge of the Σ protocol; therefore, we can simulate the protocol's view by guessing the proof challenges without breaking the hiding property of the commitment scheme. Due to this proof technique, the number of corrupted voters t should be polynomially related to the security parameter λ in a certain way; while the total number of voters n can be any function that is $poly(\lambda)$ (as long as the correctness requirement is fulfilled, cf. theorem 3). We emphasize that given a specific n, our system can support privacy for any desired number of adversarial voters $t < n$ (as long as a suitably large security parameter λ is used).

Theorem 5 *Assume there exists a constant $c, 0 < c < 1$ such that for any 2^{λ^c}-time adversary $\mathcal{A}$, the advantage of breaking the hiding property of the commitment scheme is $\mathsf{Adv}_{\mathsf{hide}}(\mathcal{A}) = \mathsf{negl}(\lambda)$. Let $t = \lambda^{c'}$ for any constant $c' < c$. For any constant $m \in \mathbb{N}$ and $n = poly(\lambda)$, The e-voting system described in Section 3.5 is t-private with respect to the privacy game $G^{\mathcal{A},\mathcal{S}}_{t\text{-priv}}(1^\lambda, n, m)$.*

Proof: To prove our claim, we will explicitly construct a simulator $\mathcal{S}$ such that we can convert any adversary $\mathcal{A}$ who can win the privacy game $G^{\mathcal{A},\mathcal{S}}_{t\text{-priv}}(1^\lambda, n, m)$ a non-negligible probability to an adversary $\mathcal{B}$ who can break the commitment hiding assumption within $poly(\lambda) \cdot 2^t \ll 2^{\lambda^c}$ time.

Recall that in the privacy game $G^{\mathcal{A},\mathcal{S}}_{t\text{-priv}}(1^\lambda, n, m)$ the challenger $\mathcal{C}$ is maintaining a coin $b \in \{0,1\}$ and always uses the candidate selection $\mathcal{U}^b_\ell$ in the **Cast** protocol. Note when $n - t < 2$ (i.e. the number of honest voters is strictly less than 2), the simulator $\mathcal{S}$ simply outputs the view of the real **Cast** protocol. It is easy to see that, by definition, the adversary $\mathcal{A}$ loses the voter privacy game $G^{\mathcal{A},\mathcal{S}}_{t\text{-priv}}(1^\lambda, n, m)$ unconditionally. When $n - t \geq 2$, consider the following simulator $\mathcal{S}$. At the beginning of the experiment, $\mathcal{S}$ flips a coin $b' \in \{0,1\}$. For each honest voter V_ℓ, $\mathcal{S}$ receives $view_\ell = (\text{Pub}, s_\ell, \mathcal{U}^b_\ell, \alpha_\ell)$ and the candidate selections $\langle \mathcal{U}^0_\ell, \mathcal{U}^1_\ell \rangle$. If $\mathcal{U}^b_\ell = \mathcal{U}^{b'}_\ell$, $\mathcal{S}$ outputs the simulated view $view'_\ell = view_\ell$. If $\mathcal{U}^b_\ell \neq \mathcal{U}^{b'}_\ell$, $\mathcal{S}$ produces a fake s'_ℓ by switching the vote-codes for candidate selections $\mathcal{U}^b_\ell$ and $\mathcal{U}^{b'}_\ell$, i.e. replacing $\left(C^{(a_\ell)}_{\ell,j_1}, \mathcal{U}^b_\ell\right)$ and $\left(C^{(a_\ell)}_{\ell,j_2}, \mathcal{U}^{b'}_\ell\right)$ with $\left(C^{(a_\ell)}_{\ell,j_2}, \mathcal{U}^b_\ell\right)$ and $\left(C^{(a_\ell)}_{\ell,j_1}, \mathcal{U}^{b'}_\ell\right)$. $\mathcal{S}$ then outputs $view'_\ell = (\text{Pub}, s'_\ell, \mathcal{U}^{b'}_\ell, \alpha_\ell)$.

Define $\mathsf{Adv}_{G_i,G_j}(\mathcal{A}) := \frac{1}{2}\left|\Pr[\mathcal{A} = 1|G_i] - \Pr[\mathcal{A} = 1|G_j]\right|$. Consider the following sequence of games from G_0 to G_5.

Game G_0: The actual game $G^{\mathcal{A},\mathcal{S}}_{t\text{-priv}}(1^\lambda, n, m)$, where the challenger uses $\mathcal{U}^b_\ell$ in the **Cast** protocol and the above simulator $\mathcal{S}$ is invoked when $b = 1$.

By definition, $\mathsf{Adv}_{G^{\mathcal{A},\mathcal{S}}_{t\text{-priv}}(1^\lambda,n,m),G_0}(\mathcal{A}) = 0$.

Game G_1: Game G_1 is the same as Game G_0 except the following. At the beginning of the experiment, the challenger $\mathcal{C}$ generates a set of coins $\{a_\ell\}_{\ell=1}^n$ uniformly at random. During the experiment, for each voter $V_\ell \in \mathcal{V}$, the adversary $\mathcal{A}$ first chooses whether V_ℓ is corrupted. If V_ℓ is not corrupted, $\mathcal{C}$ uses a_ℓ in the **Cast** protocol to vote on behave of V_ℓ according to $\mathcal{U}^b_\ell$; otherwise, $\mathcal{C}$ sends s_ℓ to the adversary $\mathcal{A}$ and interacts with $\mathcal{A}$ in the **Cast** protocol, playing the role of EA and BB. Let $\hat{a}_\ell$ be the coin used by the corrupted voter $\hat{V}_\ell \in \mathcal{V}_{\mathsf{corr}}$ in the **Cast** protocol execution. The experiment aborts and start over if there exists one corrupted voter's coin $\hat{a}_\ell \neq a_\ell$.

It is easy to see that, no matter how $\mathcal{V}_{\mathsf{corr}}$ is chosen, it requires (expected) at most 2^t attempts to guess all the $\hat{a}_\ell$ correctly given $|\mathcal{V}_{\mathsf{corr}}| \leq t$. On the other hand, when $\mathcal{C}$ does not abort, the view of Game G_1 is identical to that of Game G_0. Hence, $\mathsf{Adv}_{G_0,G_1}(\mathcal{A}) = 0$.

Game G_2: Game G_2 is the same as Game G_1 except the following. The challenger $\mathcal{C}$ computes a set of commitments $\{E_j\}_{j=1}^m$, where $E_j = \mathsf{Com}_{\mathsf{ck}}((n+1)^{j-1}; r_j)$ with fresh randomizer $r_j \in \mathbb{Z}_q$.

For each ballot, for $a_\ell \in \{0,1\}$, $\mathcal{C}$ permutes and re-randomizes $\{E_j\}_{j=1}^m$ to produce the commitments $\left\{E_{\ell,j}^{(a_\ell)}\right\}_{j=1}^m$ instead of committing them from scratch as follows.

- Pick a random permutation $\pi_\ell^{(a_\ell)}$ over $[m]$.
- For $j \in [1,m]$,
  - Pick a random $r_{\ell,j}^{(a_\ell)} \leftarrow \mathbb{Z}_q$;
  - Set $E_{\ell,j}^{(a_\ell)} = E_{\pi_\ell^{(a_\ell)}(j)} \cdot \mathsf{Com}_{\mathsf{ck}}(0; r_{\ell,j}^{(a_\ell)})$;

It is straightforward that the view of Game G_2 is identical to that of Game G_1, as the distributions of all the commitments are the same. Hence, $\mathsf{Adv}_{G_1,G_2}(\mathcal{A}) = 0$.

Game G_3: Game G_3 is the same as Game G_2 except the following. $\mathcal{C}$ randomly selects $\ell^* \in [n]$ and guesses the tally vector $(t_1,\ldots,t_m)$. Denote $T = \sum_{i=1}^m t_i \cdot (n+1)^{i-1}$. $\mathcal{C}$ aborts if either of the following two events occur: (i) $\mathcal{A}$ corrupts V_{ℓ^*} and then does not let V_{ℓ^*} submit a vote; (ii) the guessed T is wrong. When $\mathcal{C}$ does not abort, it generates the challenge(s) ρ using the guessed voters' coins $\{a_\ell\}_{\ell\in[n]}$ in Game G_1, and then replaces all the real Σ protocols with their simulated transcripts.

The probability $\mathcal{A}$ corrupts V_{ℓ^*} and then does not let V_{ℓ^*} submit a vote is at most $\frac{t}{n}$ (Namely all the corrupted voters abort). Besides, the probability the $\mathcal{C}$ guesses T correctly is at least $\frac{1}{n(n+1)^{m-1}}$. Hence, it requires (expected) at most $\frac{n^2(n+1)^{m-1}}{n-t}$ attempts to get both events occur. On the other hand, when $\mathcal{C}$ does not abort, according to Lemma 1, for each Σ protocol, the adversary can distinguish the simulated transcript from a real one with advantage at most $\log m \cdot \mathsf{Adv}_{\mathsf{hide}}(\mathcal{A})$. There are $2nm$ simulated Σ protocols, so by union bound we have $\mathsf{Adv}_{G_2,G_3}(\mathcal{A}) \leq 2nm \log m \cdot \mathsf{Adv}_{\mathsf{hide}}(\mathcal{A})$.

Game G_4: Game G_4 is the same as Game G_3 except the following. At the beginning of the experiment, $\mathcal{C}$ replaces the set of commitments $\{E_j\}_{j=0}^{m-1}$ in Game G_3 with commitments of 0. Let T and ℓ^* be the ones guessed in Game G_3. For all $\ell \in [n] \wedge \ell \neq \ell^*$, $\mathcal{C}$ produces $\left\{E_{\ell,j}^{(a_\ell)}\right\}_{j=1}^m$ by re-randomizing $\{E_j\}_{j=1}^m$ as in Game G_3; $\mathcal{C}$ replaces all the commitments $\left\{E_{\ell^*,j}^{(a_{\ell^*})}\right\}_{j=1}^m$ with fresh commitments of T, i.e. $\mathsf{Com}_{\mathsf{ck}}(T; R_j)$, where $R_j \leftarrow \mathbb{Z}_q$ are chosen uniformly at random.

We now show that the view of Game G_4 is indistinguishable to that of Game G_3 by reduction. Suppose the adversary $\mathcal{B}$ is playing the hiding game of the underlying commitment scheme. On receiving ck, $\mathcal{B}$ queries two message $m_0 = 0$ and $m_1 = 1$. Given $E = \mathsf{Com}_{\mathsf{ck}}(m_b; *)$, $\mathcal{B}$ needs to guess b. $\mathcal{B}$ plays as role of the challenger in the game where $\mathcal{A}$ is trying to distinguish between G_3 and G_4. (Assume that $\mathcal{A}$ outputs 1 if she thinks she is in G_3 and outputs 0 if she thinks she is in G_4.) For $j \in [m]$, $\mathcal{B}$ sets $E_j = E^{(n+1)^{j-1}}$. For $\ell \in [n] \wedge \ell \neq \ell^*$, $\mathcal{B}$ produces $\left\{E_{\ell,j}^{(a_\ell)}\right\}_{j=1}^m$ by re-randomizing $\{E_j\}_{j=1}^m$ as in Game G_3. For $j \in [m]$, $\mathcal{C}$ picks $R_j \leftarrow \mathbb{Z}_q$ at random and sets

$$E_{\ell^*,j}^{(a_{\ell^*})} = \mathsf{Com}_{\mathsf{ck}}(T, R_j)/(E_j^{t_j-1} \cdot \prod_{i=1, i\neq j}^{m} E_i^{t_i}) \ .$$

Denote abort as either the events: (i) $\mathcal{A}$ corrupts V_{ℓ^*} and then does not let V_{ℓ^*} submit a vote; (ii) the guessed T is wrong. Clearly, if E commits to 1, then the produced view is identical to G_3; if E commits to 0, then the produced view is identical to G_4. Except in Game G_4, there are commitments of T in Pub, so the adversary $\mathcal{A}$ might be able to intentionally make the abort abort event occurs with a higher probability. To address this, $\mathcal{C}$ maintains a counter, and $\mathcal{C}$ increases the counter by 1 each time abort occurs. Denote halt as the event where abort continuously occurs $\frac{n^3(n+1)^{m-1}}{n-t}$ times, and when halt occurs, $\mathcal{C}$ outputs 0 in the commitment hiding game; otherwise, $\mathcal{B}$ forwards the bit that $\mathcal{A}$ outputs.

The probability $\mathcal{B}$ wins the hiding game is

$$\begin{aligned}
\Pr[\mathcal{B}=b] &= \Pr[\mathcal{B}=b|\text{halt}]\cdot\Pr[\text{halt}]+\Pr[\mathcal{B}=b|\neg\text{halt}]\cdot\Pr[\neg\text{halt}]\\
&= \Pr[b=0|\text{halt}]\cdot\Pr[\text{halt}]+\frac{1}{2}\Pr[\mathcal{B}=0|b=0\wedge\neg\text{halt}]\cdot\Pr[\neg\text{halt}]\\
&\quad +\frac{1}{2}\Pr[\mathcal{B}=1|b=1\wedge\neg\text{halt}]\cdot\Pr[\neg\text{halt}]\\
&= (1-\Pr[b=1|\text{halt}])\cdot\Pr[\text{halt}]+\frac{1}{2}\Pr[\mathcal{A}=1|G_3]\cdot\Pr[\neg\text{halt}]\\
&\quad +\frac{1}{2}\Pr[\mathcal{A}=0|G_4]\cdot\Pr[\neg\text{halt}]\\
&\geq (1-\frac{n-t}{n^2(n+1)^{m-1}})^{\frac{n^3(n+1)^{m-1}}{n-t}}\cdot\Pr[\text{halt}]\\
&\quad +(\frac{1}{2}+\frac{1}{2}\Pr[\mathcal{A}=1|G_3]-\frac{1}{2}\Pr[\mathcal{A}=1|G_4])\cdot\Pr[\neg\text{halt}]\\
&\geq (1-e^{-n})\cdot\Pr[\text{halt}]+(\frac{1}{2}+\mathsf{Adv}_{G_3,G_4}(\mathcal{A}))\cdot\Pr[\neg\text{halt}]\\
&\geq \min\left(\frac{1}{2}+\mathsf{Adv}_{G_3,G_4}(\mathcal{A}),1-e^{-n}\right).
\end{aligned}$$

Since we assume that no adversary $poly(\lambda)$-time $\mathcal{A}$ can win the hiding game with non-negligible advantage, we have $\frac{1}{2}+\mathsf{Adv}_{G_3,G_4}(\mathcal{A})\leq 1-e^{-n}$; hence, $\mathsf{Adv}_{G_3,G_4}(\mathcal{A})\leq\mathsf{Adv}_{\mathsf{hide}}(\mathcal{A})$.

Game G_5**:** Game G_5 is the same as Game G_4 except the following. For each honest voter $V_\ell\in\tilde{\mathcal{V}}$, $\mathcal{C}$ picks $\tilde{\mathcal{U}}_\ell$ at random, and uses $\tilde{\mathcal{U}}_\ell$ in the **Cast** protocol, ignoring the adversary's $\langle\mathcal{U}^0_\ell,\mathcal{U}^1_\ell\rangle$. Regardless the coin b, $\mathcal{C}$ always uses the simulator $\mathcal{S}$ to transform the view $view_\ell=(\text{Pub},s_\ell,\tilde{\mathcal{U}}_\ell,\alpha_\ell)$ to $view'_\ell=(\text{Pub},s'_\ell,\mathcal{U}^{b'}_\ell,\alpha_\ell)$.

It is obvious that the view of G_5 is identical to the view of G_4, so $\mathsf{Adv}_{G_4,G_5}(\mathcal{A})=0$. Notice that all the vote-codes are generated at random and all the commitments in each ballots commit to the same value (0 or T). Moerover, since the view of G_5 does not depend on the challenger's coin b, we have the probability that $\mathcal{A}$ guess b correctly is $\Pr[\mathcal{A}=1|G_5]=\frac{1}{2}$.

To sum up, the total running time of our reduction is $poly(\lambda)\cdot 2^t$ and we have

$$\begin{aligned}
\Pr[G^{\mathcal{A},\mathcal{S}}_{t\text{-priv}}(1^\lambda,n,m)=1] &= \Pr[\mathcal{A}=1|G_5]+\sum_{i=1}^{5}\mathsf{Adv}_{G_{i-1},G_i}(\mathcal{A})\\
&\leq \frac{1}{2}+(2nm\log m+1)\cdot\mathsf{Adv}_{\mathsf{hide}}(\mathcal{A})\\
&= \frac{1}{2}+\mathsf{negl}(\lambda).
\end{aligned}$$

□

5 Implementation and Performance

Similar to Helios, our system is an open source web-based public auditable e-voting system. Its web interface is written in *Django* language and *Twitter Bootstrap* CSS and javascript framework [11] are used for better appearance. The system consists of three main components (servers): election authority (EA), voter bulletin board (VBB), and audit bulletin board (ABB). All the transmissions between those components utilize *Google's Protocol Buffers* [28] to encode the structured data. To create an election, the election committee needs to login to the EA site and define an election by specifying election question, options, election period, total number of ballots, ballot distribution methods, and trustees, etc. Once an election is created, the EA server will prepare the ballots at the backend. The elliptic curve ElGamal and the corresponding ZK proof are implemented in C++, using *Multi-precision Integer and Rational Arithmetic C/C++ Library* (MIRACL) crypto SDK [40]. Based on the number of voters n and the number of options m, the system will dynamically select an elliptic curve domain parameter from one of the NIST p192, p224, p256, p384, p521 curves, such that $n \cdot (n+1)^{m-1}$ can fit in the message space. We also use the standard point compression technique: a point on the curve is represented by its x coordinate together with the least significant bit of its y coordinate. After that, the EA distributes the ballots to the voters, sends tally keys to the trustees and pushes the audit data to the ABB site. One very important feature of our e-voting system is its no voter-side-crypto design principle, so we keep the VBB site clean and simple. Hence, the voters are able to use some commercial off-the-shelf (COT) smartphones to cast a vote. In the tally phase, the trustees based on the bulletin board information compute their tally shares, using their secret keys. The *Stanford Javascript Crypto Library (SJCL)* [47] is employed to facilitate the necessary local cryptographic computation at the trustee side. Finally, the election result will be displaced on both the VBB and ABB sites. In terms of performance, the cryptographic computation in the election preparation step denominates the entire election process. The benchmark results in Table 1 are obtained on a Debian server with Intel i7-4700HQ 2.4 GHz, 16GB RAM.

n	m	Curve	Preparation Time
1000	2	p192	21 seconds
1000	5	p192	91 seconds
1000	10	p192	220 seconds
10000	2	p192	3.5 minutes
10000	5	p192	15 minutes
10000	10	p192	36 minutes

Table 1: Election Preparation Time Benchmark

References

[1] Ben Adida. Helios: Web-based open-audit voting. In *USENIX Security*, 2008.

[2] B. Barak, G. Kindler, R. Shaltiel, B. Sudakov, and A. Wigderson. Simulating independence: New constructions of condensers, ramsey graphs, dispersers, and extractors. *J. ACM*, 57(4):20:1–20:52, May 2010.

[3] Carsten Baum, Ivan Damgård, and Claudio Orlandi. Publicly auditable secure multi-party computation. In *SCN 2014 Proceedings*, 2014.

[4] Donald Beaver. Plug and play encryption. In *CRYPTO'97*, pages 75–89, 1997.

[5] Josh Benaloh. Simple verifiable elections. In *USENIX*, 2006.

[6] Josh Cohen Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *STOC*, 1994.

[7] Josh Cohen Benaloh and Moti Yung. Distributing the power of a government to enhance the privacy of voters (extended abstract). In *PODC*, 1986.

[8] David Bernhard, Véronique Cortier, Olivier Pereira, Ben Smyth, and Bogdan Warinschi. Adapting helios for provable ballot privacy. In *ESORICS*, 2011.

[9] David Bernhard, Olivier Pereira, and Bogdan Warinschi. How not to prove yourself: Pitfalls of the Fiat-Shamir heuristic and applications to Helios. In *Asiacrypt*, 2012.

[10] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *STOC*, 1988.

[11] Bootstrap. Twitter Bootstrap. `http://getbootstrap.com/`, 2013.

[12] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–88, 1981.

[13] David Chaum. Surevote: Technical overview. In *Proceedings of the Workshop on Trustworthy Elections*, WOTE, 2001.

[14] David Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security & Privacy*, 2(1):38–47, 2004.

[15] David Chaum, Richard Carback, Jeremy Clark, Aleksander Essex, Stefan Popoveniuc, Ronald L. Rivest, Peter Y. A. Ryan, Emily Shen, Alan T. Sherman, and Poorvi L. Vora. Scantegrity II: end-to-end verifiability by voters of optical scan elections through confirmation codes. *IEEE TIFS*, 4(4):611–627, 2009.

[16] Benoît Chevallier-Mames, Pierre-Alain Fouque, David Pointcheval, Julien Stern, and Jacques Traoré. On some incompatible properties of voting schemes. In *Towards Trustworthy Elections*, 2010.

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

[18] Josh D. Cohen and Michael J. Fischer. A robust and verifiable cryptographically secure election scheme (extended abstract). In *FOCS*, 1985.

[19] Josh D. Cohen and Michael J. Fischer. A robust and verifiable cryptographically secure election scheme (extended abstract). In *FOCS*, 1985.

[20] United States Election Assistance Commission. Voluntary voting systems guidelines, 2005.

[21] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO*, 1994.

[22] Ronald Cramer, Matthew K. Franklin, Berry Schoenmakers, and Moti Yung. Multi-autority secret-ballot elections with linear work. In *EUROCRYPT*, 1996.

[23] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. *ETT*, 8(5):481–490, 1997.

[24] Olivier de Marneffe, Olivier Pereira, and Jean-Jacques Quisquater. Simulation-based analysis of E2E voting systems. In *Frontiers of Electronic Voting*, 2007.

[25] Stéphanie Delaune, Steve Kremer, and Mark Ryan. Verifying privacy-type properties of electronic voting protocols. *J. of Computer Security*, 17(4):435–487, 2009.

[26] Danny Dolev, Michael J Fischer, Rob Fowler T, Nancy A Lynch, and H. Raymond Strong. An efficient algorithm for byzantine agreement without authentication. *Information and Control*, 52:257–274, 1982.

[27] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *EUROCRYPT'15*. Springer, 2015.

[28] Google. Protocol Buffers. `https://code.google.com/p/protobuf/`, 2013.

[29] Jens Groth. Evaluating security of voting schemes in the universal composability framework. In *ACNS'04*, pages 46–60, 2004.

[30] Eitan M. Gurari. *Introduction to the theory of computation*. Computer Science Press, 1989.

[31] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. *IACR Cryptology ePrint Archive*, 2002:165, 2002.

[32] Jesse Kamp and David Zuckerman. Deterministic extractors for bit-fixing sources and exposure-resilient cryptography. *SIAM J. Comput.*, 36(5):1231–1247, 2006.

[33] Aggelos Kiayias, Michael Korman, and David Walluck. An internet voting system supporting user privacy. In *ACSAC*, 2006.

[34] Steve Kremer, Mark Ryan, and Ben Smyth. Election verifiability in electronic voting protocols. In *ESORICS*, pages 389–404, 2010.

[35] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Accountability: Definition and relationship to verifiability. *IACR Cryptology ePrint Archive*, 2010:236, 2010.

[36] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. A game-based definition of coercion-resistance and its applications. In *CSF*, pages 122–136, 2010.

[37] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Verifiability, privacy, and coercion-resistance: New insights from a case study. In *IEEE Symposium on Security and Privacy*, pages 538–553. IEEE Computer Society, 2011.

[38] David Lichtenstein, Nathan Linial, and Michael E. Saks. Imperfect random sources and discrete controlled processes. In *STOC*, pages 169–177, 1987.

[39] Silvio Micali, Rafael Pass, and Alon Rosen. Input-indistinguishable computation. In *FOCS*, pages 367–378. IEEE Computer Society, 2006.

[40] MIRACL. Multi-precision Integer and Rational Arithmetic C/C++ Library. `http://www.certivox.com/miracl/`, 2013.

[41] Tal Moran and Moni Naor. Receipt-free universally-verifiable voting with everlasting privacy. In *CRYPTO*, pages 373–392, 2006.

[42] C. Andrew Neff. Practical high certainty intent verification for encrypted votes. Votehere, Inc. whitepaper, 2004.

[43] Stefan Popoveniuc, John Kelsey, Andrew Regenscheid, and Poorvi Voral. Performance requirements for end-to-end verifiable elections. In *EVT/WOTE*, 2010.

[44] Ran Raz. Extractors with weak random seeds. In *STOC*, 2005.

[45] Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme - a practical solution to the implementation of a voting booth. In *EUROCRYPT*, 1995.

[46] Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980.

[47] SJCL. Stanford Javascript Crypto Library. `http://crypto.stanford.edu/sjcl/`, 2013.

[48] Dominique Unruh and Jörn Müller-Quade. Universally composable incoercibility. *IACR Cryptology ePrint Archive*, 2009:520, 2009.

[49] Filip Zagórski, Richard Carback, David Chaum, Jeremy Clark, Aleksander Essex, and Poorvi L. Vora. Remotegrity: Design and use of an end-to-end verifiable remote voting system. In *ACNS*, 2013.

[50] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *EUROSAM*, 1979.