# (Universal) Unconditional Verifiability in E-Voting without Trusted Parties

Vincenzo Iovino*, Alfredo Rial†, Peter B. Rønne†, Peter Y. A. Ryan†

*University of Salerno, Italy
Email: vinciovino@gmail.com

†SnT, University of Luxembourg
Email: {alfredo.rial, peter.roenne, peter.ryan}@uni.lu

***Abstract*—In e-voting protocols, cryptographers must balance usability with strong security guarantees, such as privacy and verifiability. In traditional e-voting protocols, privacy is often provided by a trusted authority that learns the votes and computes the tally. Some protocols replace the trusted authority by a set of authorities, and privacy is guaranteed if less than a threshold number of authorities are corrupt. For verifiability, stronger security is demanded. Typically, corrupt authorities that try to fake the tally result must always be detected.**

**To provide verifiability, many e-voting protocols use Non-Interactive Zero-Knowledge proofs (NIZK). Thanks to their non-interactive nature, NIZK allow anybody, including third parties that do not participate in the protocol, to verify the correctness of the tally. Therefore, NIZK can be used to obtain universal verifiability. Additionally, NIZK also improve usability because they allow voters to cast a vote non-interactively.**

**The disadvantage of NIZK is that their security is based on setup assumptions such as the common reference string (CRS) or the random oracle model. The former requires a trusted party to generate a CRS. The latter, though a popular model for secure protocol design, has been shown to be unsound.**

**We address the design of e-voting protocols that provide verifiability without any trust assumptions. We show that Non-Interactive Witness-Indistinguishable proofs can be used for this purpose. Our e-voting protocols are private under the Decision Linear assumption, while *perfect* individual verifiability, i.e. a fake tally is detected with probability 1, holds unconditionally. *Perfect* universal verifiability requires a trusted public bulletin board. We remark that our definition of verifiability does not consider eligibility or end-to-end verifiability. First, we present a general construction that supports any tally function. Then, we show how to efficiently instantiate it for specific types of elections through Groth-Sahai proofs.**

## I. Introduction

The parties participating in a standard e-voting protocol are multiple voters and one authority. First, the authority computes a key pair, keeps the secret key and publishes the public key. A voter computes a ballot on input the public key of the authority and her intended vote and sends the ballot to a public bulletin board (PBB), which records it in an entry associated with that voter. In case of abstention, a special symbol $\perp$ is recorded on the PBB. The authority uses its secret key to compute the tally on input all the ballots on the PBB, which could possibly be $\perp$ in case of abstention. Finally, the correctness of the tally can be checked by running a verification algorithm.[1]

E-voting protocols must provide two security properties: privacy and verifiability. Privacy should protect the secrecy of the votes. Verifiability should prevent a corrupt authority from faking the tally.

Privacy protection assumes the existence of a trusted authority in many e-voting systems [1]–[8]. As for schemes that distribute the trust among several authorities, privacy protection still requires that not all of the authorities are corrupt. However, *verifiability* (also called integrity) should be ensured even if the authorities are corrupt.

Many e-voting systems make use of Non-Interactive Zero-Knowledge Proofs (NIZK) [9]–[13] to provide verifiability. NIZK must provide two properties: soundness and zero-knowledge (ZK). Soundness prevents a corrupt prover from proving a false statement, i.e., a statement for which no witness exists. ZK ensures that the verifier does not learn any information about the witness.

ZK is defined following the simulation paradigm, i.e., it requires the existence of a simulator that computes a valid proof without knowledge of the witness. However, if such a simulator existed, soundness would not hold. This apparent contradiction is solved by resorting to trust assumptions like the Common Reference String (CRS) model [9]. In the CRS model, a trusted party generates a CRS that is used by both provers and verifiers. The simulator is given the additional power of computing the CRS. Thanks to that, the simulator knows trapdoor information that allows it to simulate proofs for all statements.

For some applications of NIZK, the CRS model is not problematic. For instance, in IND-CCA public key encryption schemes [13]–[15], ZK does not need to hold for the receiver of ciphertexts because the receiver must be able to decrypt anyway. Therefore, the CRS is computed by the receiver, while NIZK are computed by the sender. However, in e-voting, the authority cannot compute the CRS because it must compute proofs that show the correctness of the tally.

[1] In this description we skipped some details (e.g., eligibility and authentication) that are not relevant to our setting. See below for more discussion.

An alternative to the CRS model is the Random Oracle (RO) model [16]. The RO model assumes that a perfect random function is available to all the parties. NIZK that use the RO model are constructed following the Fiat-Shamir heuristic [17]. To prove that NIZK constructed following this heuristic are ZK, we need programmability of the RO, i.e., the ability of the simulator to change the input/output of the RO.[2]

To compute a proof, in practice, the prover replaces the RO by some "secure" hash function. Therefore, this hash function must be chosen honestly for ZK to hold. Consequently, all the parties must trust the implementation of a concrete hash function (e.g., SHA-3 [18]). We note that a hash function could have been designed in a malicious way (e.g., "programmed" like in the simulation) to allow the computation of a proof for a false statement. Moreover, even when programmability is not needed, the RO methodology has been shown to be unsound [19]. Further problems are known regarding the programmability of the RO in the context of NIZK [20]–[22]. The current techniques to avoid the need of programmability resort to the CRS model [23]–[26]. This motivates our main question: is it possible to design a verifiable e-voting scheme without trust assumptions (like CRS and RO)?

Lipmaa [27] asks whether Non-Interactive Witness Indistinguishable Proofs (NIWI) can be used to replace NIZK: "Moreover, we think that the CRS model is almost realistic, but it would still be desirable to do without it. The implication of non-interactive witness-indistinguishable protocols to the e-voting is something that must still be studied." NIWI can be constructed without using any trust assumptions [28]–[31].[3] Recently, Bellare, Fuchsbauer and Scafuro [33] started the study of security of NIZKs in face of public parameter subversion. They showed the impossibility of attaining subversion soundness while retaining ZK, thus justifying our need of sidestepping NIZKs. Subsequent works show that there is an increasing interest in studying security under parameter subversion [34], [35].

NIWI is a non-interactive proof/argument system that provides weaker security guarantees in comparison to NIZK. While NIZK ensure that a proof does not reveal any information about the witness, NIWI only guarantee that, for any two witnesses $w_1$ and $w_2$ for the same statement, a proof computed on input $w_1$ is computationally indistinguishable from a proof computed on input $w_2$. Note that this notion only makes sense for languages with multiple witnesses for each statement, which is not always the case.

To our knowledge, it was not known how to use NIWI to construct an e-voting scheme (eVote, in short) that is both private and verifiable. Usually, it is very difficult to use NIWI because of its weaker security guarantee. Nonetheless, inspired by a recent result on functional encryption [36], [37] of Badrinarayanan, Goyal, Jain and Sahai [38], surprisingly we are able to use NIWI to answer our main question affirmatively.

### A. Our Results

First, we define the correctness, privacy and verifiability properties for an eVote. Then we propose a private and verifiable eVote that supports any tally function (representable as a polynomial-sized Boolean circuit). Our eVote uses as building blocks a NIWI proof system, a public-key encryption scheme with perfect correctness and unique secret key, and a perfectly-binding commitment scheme. It can be instantiated by using just bilinear groups [39], [40]. For instance, we can instantiate our construction with the NIWI of Groth, Ostrovsky and Sahai [28] and the Decision Linear (DLIN) encryption scheme of Boneh *et al.* [41].

Our construction provides universal verifiability, i.e. parties that do not participate in the elections can verify the correctness of the tally, under the only assumption of a trusted public bulletin board (PBB). Without a trusted PBB, universal verifiability degrades to an individual verifiability notion. Therefore, our construction achieves *perfect* individual verifiability, i.e., individual verifiability is achieved without trusted parties and without any assumption, while universal verifiability needs a trusted PBB. Our definition of verifiability does not consider eligibility. In Section II-D, we discuss how eligibility can be attained. Our definition of verifiability does not consider end-to-end verifiability. Concretely, it does not consider the case in which an adversarial voting device can modify the vote of an honest voter. The DLIN assumption is only needed to prove that our eVote fulfills the privacy property. This assumption is well-studied and falsifiable [42]. This is a key point of our results because otherwise one could just claim that an eVote in the RO model is secure when instantiated with any hash function.

Our eVote supports any tally function representable as a polynomial-sized Boolean circuit. The drawback is that the computation has to be expressed as a Boolean circuit as well and, though all algorithms run in probabilistic polynomial time, the overall performances may be prohibitive in practice. Nevertheless, in Appendix E, we provide an efficient instantiation of our eVote for the concrete case of the sum function over a binary domain (i.e., a referendum). This instantiation uses Groth-Sahai proofs [32], [43], whose security can also be proven under the DLIN assumption. We provide an efficiency analysis to attest the practicality of our instantiation.

In Section V, we outline how to adapt our construction for general functions to a model with multiple authorities. In this model, the tally evaluation algorithm is run by a set of authorities and the privacy property must hold if at least one authority is honest. (As this is not the main focus of our work, we do not present formal definitions and details for its construction.) An important advantage of our construction is that no interaction among the authorities is required. In this respect, our techniques completely diverge

[2]We remark that it is not known how to obtain efficient NIZK *proofs* in the RO model. In fact, NIZK systems that use the Fiat-Shamir heuristic only satisfy soundness against polynomial-time provers, and thus are not statistically sound.

[3]Note that, in the literature, there are both NIWI in the CRS model, like the ones of Groth and Sahai [32], and one-message NIWI without CRS (see the citations above). Henceforth, unless specified otherwise, we denote by NIWI the (one-message) variant without CRS, and in particular we refer to the NIWI for CircuitSat of Groth *et al.* [28].

from previous approaches to the problem and may be of independent interest. We stress that the multi-string model of Groth and Ostrovsky [44], though conceptually appealing in this scenario, fails to provide a solution.

In this work, we use cryptographic primitives to demonstrate the achievability of perfect verifiable e-voting systems. However, we are not concerned about usability and "human-friendly" verifiability, as dealt with in [45]–[47]. Moreover, we only consider traditional e-voting systems and hence we neglect other approaches [48]–[53].

**Organization.** In Section II, we define an eVote and its verifiability and privacy properties. In Section III we present the building blocks we use in our construction. In Section IV, we present our construction for an eVote. In Section V, we discuss a construction with multiple authorities. In Section VI we discuss related work. In Section VII we discuss future directions in cryptography and e-voting that our work opens up.

## II. Definitions

### A. Notation

A *negligible* function $\mathsf{negl}(k)$ is a function that is smaller than the inverse of any polynomial in $k$ (from a certain point and on). We denote by $[n]$ the set of numbers $\{1,\ldots,n\}$. If $S$ is a finite set, we denote by $a \leftarrow S$ the process of setting $a$ equal to a uniformly chosen element of $S$. With a slight abuse of notation, we assume the existence of a special symbol $\perp$ that does not belong to $\{0,1\}^\star$.

If $A$ is an algorithm, then $A(x_1,x_2,\ldots)$ denotes the probability distribution of the output of $A$ when $A$ is run on input $(x_1,x_2,\ldots)$ and randomly chosen coin tosses. Instead, $A(x_1,x_2,\ldots;r)$ denotes the output of $A$ when run on input $(x_1,x_2,\ldots)$ and (sufficiently long) coin tosses $r$. All algorithms, unless explicitly noted, are probabilistic polynomial time (PPT) and all adversaries are modeled by non-uniform PPT algorithms.

If $A$ is a PPT algorithm, we say that $y \in A(x)$ iff there exists a random value $r$ such that $y = A(x;r)$; in that case, we say that $y$ is in the range of $A(x)$. If $E$ is an event in a probability space, $\bar{E}$ denotes its complement.

The following definition is used in the definition of verifiability. Essentially, it states that a tally $y$ is compatible with votes $z_1,\ldots,z_k$ if the latter values are in its pre-image.

*Definition 2.1:* Given a function $F(x_1,\ldots,x_n) : A^n \rightarrow B$, we say that a value $y \in B$ is compatible with $z_1,\ldots,z_k \in A$ at indices $i_1,\ldots,i_k \in [N]$ if $y$ is in the range of the restriction $F_{|C_{z_1,\ldots,z_k,i_1,\ldots,i_k}}$ of $F$ to $C_{z_1,\ldots,z_k,i_1,\ldots,i_k} \stackrel{\triangle}{=} \{(x_1,\ldots,x_n) | \forall j \in [k], x_{i_j} = z_j\}$.

### B. E-Voting Schemes

An eVote is parameterized by the tuple $(N,\mathcal{M},\Sigma,F)$. The natural number $N > 0$ is the *number of voters*. The set $\mathcal{M}$ is the *domain of valid votes*. The set $\Sigma$ is the *range of possible results*. The function $F : (\mathcal{M} \cup \{\perp\})^N \rightarrow \Sigma \cup \{\perp\}$ is the *tally function*. We allow the tally function to take as input the special symbol $\perp$, which denotes either an abstention, an invalid ballot or a blank vote[4], and to output $\perp$ to indicate an error. We require that the tally function outputs an error on input a sequence of strings iff all the strings are equal to $\perp$. Formally, the tally function is defined as follows.

*Definition 2.2:* [Tally function] A function $F$ is a tally function if there exists a natural number $N > 1$, and sets $\mathcal{M},\Sigma \subset \{0,1\}^\star$ such that the domain of $F$ is $(\mathcal{M} \cup \{\perp\})^N$, the range is $\Sigma \cup \{\perp\}$ and for all strings $m_1,\ldots,m_N \in \mathcal{M} \cup \{\perp\}$, it holds that $F(m_1,\ldots,m_N) = \perp$ iff $m_1 = \perp,\ldots,m_N = \perp$.

We use a simple e-voting model with a single authority. In Section V, we outline how to adapt our construction to a model with multiple authorities.

*Definition 2.3:* [E-voting Scheme] A $(N,\mathcal{M},\Sigma,F)$-*e-voting scheme* $\mathsf{EVOTE}$ for number of voters $N > 1$, domain of valid votes $\mathcal{M}$, range of possible results $\Sigma$ and tally function $F : (\mathcal{M} \cup \{\perp\})^N \rightarrow \Sigma \cup \{\perp\}$ is a tuple

$$\mathsf{EVOTE} \stackrel{\triangle}{=} (\mathsf{Setup},\mathsf{Cast},\mathsf{VerifyBallot},\mathsf{EvalTally},\mathsf{VerifyTally})$$

of 5 PPT algorithms, where VerifyBallot and VerifyTally are deterministic, that fulfill the following syntax:

1) $\mathsf{Setup}(1^\lambda)$: on input the security parameter in unary, it outputs the *public key* Pk and the *secret key* Sk.
2) $\mathsf{Cast}(\mathsf{Pk},j,v)$: on input the public key Pk, the voter identifier $j \in [N]$, and a vote $v \in \mathcal{M}$, it outputs a *ballot* Blt.
3) $\mathsf{VerifyBallot}(\mathsf{Pk},j,\mathsf{Blt})$: on input the public key Pk, the voter identifier $j \in [N]$ and a ballot Blt, it outputs a value in $\{\mathsf{OK},\perp\}$.
4) $\mathsf{EvalTally}(\mathsf{Pk},\mathsf{Sk},\mathsf{Blt}_1,\ldots,\mathsf{Blt}_N)$: on input the public key Pk, the secret key Sk, and $N$ strings that are either ballots or the special symbol $\perp$, it outputs the tally $y \in \Sigma \cup \{\perp\}$ and a proof $\gamma$ of tally correctness.
5) $\mathsf{VerifyTally}(\mathsf{Pk},\mathsf{Blt}_1,\ldots,\mathsf{Blt}_N,y,\gamma)$: on input the public key Pk, $N$ strings that are either ballots or the special symbol $\perp$, a tally $y \in \{0,1\}^\star \cup \{\perp\}$ and a proof $\gamma$ of tally correctness, it outputs a value in $\{\mathsf{OK},\perp\}$.

The voting ceremony is as follows.

- Setup phase. An authority (also called voting authority or election authority) uses algorithm Setup to compute a public key Pk and a secret key Sk.
- Voting phase. Each of the $N$ voters runs an algorithm Cast on input the voter identifier $j \in [N]$, the public key Pk and a vote $v \in \mathcal{M}$ to compute a ballot Blt. The voter sends Blt to an append-only public bulletin board (PBB).
- Tallying phase. The well-formedness of each ballot Blt published in the PBB can be publicly verified by means of an algorithm VerifyBallot. If the ballot is invalid, a new row in which the ballot is replaced by $\perp$ is appended to the PBB. Later, only the new row is used. If a voter did not cast a vote, $\perp$ is appended to the PBB.
  The authority runs algorithm EvalTally on input the public key, the secret key, and $N$ strings that represent either ballots or $\perp$ symbols appended to the PBB. EvalTally

[4]We note that our tally function can be made more general by assigning different symbols to an abstention, to an invalid ballot and to a blank vote.

outputs the tally, i.e., the result of the election, and a proof of tally correctness. The tally equals the special symbol $\perp$ to indicate an error.

- Verification phase. Algorithm $\mathsf{VerifyTally}$ takes as input the public key, a tuple of $N$ strings that represent either ballots or the special symbol $\perp$, the tally and the proof of tally correctness. $\mathsf{VerifyTally}$ outputs a value in $\{\mathsf{OK}, \perp\}$. Each participant, not necessarily a voter, can verify the correctness of the result of the election as follows. First, verify whether the ballots cast by the voters are valid using the $\mathsf{VerifyBallot}$ algorithm. Check whether the authority replaced with $\perp$ only the invalid ballots. Assign $\perp$ to any voter who did not cast her vote. After that, run the $\mathsf{VerifyTally}$ algorithm on input the public key, the $N$ strings that represent either ballots or the special symbol $\perp$, the tally and the proof of tally correctness.

An eVote must satisfy the following correctness, verifiability, and privacy properties.

### C. Definition of Correctness

Traditionally, the correctness property guarantees both (1) that the ballot verification algorithm accepts the ballots computed by the cast algorithm, and (2) that the tally verification algorithm accepts the tally and the proof computed by the tally evaluation algorithm. In the latter, the ballots taken as input by the tally evaluation algorithm are computed by the cast algorithm. Therefore, it is not guaranteed that the tally verification algorithm accepts the output of the tally evaluation algorithm when the ballots are not computed by the cast algorithm.

Consider a scheme where the ballot verification algorithm accepts any ballot. It would be possible to make such a scheme verifiable by just changing the tally verification algorithm so that it accepts $y = \perp$ only when no ballot passes the ballot verification algorithm. As can be seen, condition (1) in the definition of verifiability (cf. Def. 2.5) is fulfilled because the "if part" of the condition never holds. However, intuitively, such a scheme is incorrect. Namely, if an honest authority that runs the tally evaluation algorithm and gets $y = \perp$ (because some ballots were ill-formed), the tally verification algorithm should accept that result.

To address this issue, we add condition (2) to the definition of correctness (cf. Def. 2.4 in Fig. 1). This condition states that the tally verification algorithm must accept the output of the tally evaluation algorithm when run on input ballots that are accepted by the ballot verification algorithm (as opposed to ballots computed by the cast algorithm). We point out that in some works on definitional foundations (e.g., Bernhard *et al.* [54]) this issue has been overlooked. The verifiable eVote we design accepts $y = \perp$ only when no ballot passes the ballot verification test but, in order to fulfill this stronger correctness property, has proofs of well-formedness in the ballots.

### D. Definition of Verifiability

In our definition of verifiability (cf. Def. 2.5 in Fig. 2), we require two conditions to hold. The first condition states that, if each ballot and the proof of correctness of the tally issued by the authority are verified successfully by the respective algorithms, then each ballot $C_i$ (possibly computed on input a maliciously generated public key) must be associated with a unique message $m_i \in \mathcal{M} \cup \{\perp\}$, and the result $y$ claimed by the authority equals $F(m_1, \ldots, m_n)$.

The second one requires that, even when the adversary generates the public key, if honest voters cast a ballot that is accepted by the ballot verification algorithm, then the ballot has to be "counted". (This condition lies in some sense between correctness and verifiability as it states a requirement about honest voters.) More concretely, consider that some ballots are computed by honest voters and are accepted by the ballot verification algorithm. (These ballots could be ill-formed if they are computed on input a public key generated by the adversary.) Consider also that the remaining ballots are computed by corrupt voters. In this situation, the tally evaluation algorithm outputs a tally $y$ and a proof of correctness that, along with the public key and the ballots, is accepted by the tally verification algorithm. Then, it must be the case that the ballots sent by honest voters were counted to obtain $y$. For example, if the tally function is a sum function that sums binary votes and three honest voters cast three 1's, then the authority should not be able to claim that $y < 3$.

In this paper, for simplicity, we do not directly address issues of *eligibility*. We assume that a ballot is associated with a voter uniquely and that the adversary cannot submit a ballot on behalf of some voters. We could use e.g. a PKI and digital signatures on the ballots to prove eligibility, but such an approach is not secure against a computationally unbounded adversary. Alternatively, a private trusted setup can be used to attain eligibility without relying on a computational assumption.

### E. Definition of Privacy

Our *privacy* definition is indistinguishability-based and states that no nonuniform PPT adversary can win the following game with non-negligible advantage. The adversary receives the public key generated by a challenger and chooses two tuples of strings that encode either valid votes in the message space $\mathcal{M} \cup \{\perp\}$ or arbitrary ballots, which are cast by possibly corrupt voters. We require that the tally function outputs the same result on input any of the tuples of strings.

The challenger chooses at random one of the two tuples. The challenger runs the ballot verification algorithm on input each of the arbitrary ballots and replaces the arbitrary ballot in the tuple by $\perp$ if verification is unsuccessful. The challenger runs the cast algorithm on input each of the valid votes in the message space to compute a ballot and replaces the valid vote in the tuple by the ballot. Then the challenger computes the tally and a proof of correctness of the tally.

The new tuple, which replaces valid votes by ballots and invalid arbitrary votes by $\perp$, is given to the adversary along with a proof of the correctness of the tally. The adversary guesses which of the two tuples was chosen by the challenger.

More formally, privacy for a $(N, \mathcal{M}, \Sigma, F)$-eVote

$$\mathsf{EVOTE} \stackrel{\triangle}{=} (\mathsf{Setup}, \mathsf{Cast}, \mathsf{VerifyBallot}, \mathsf{EvalTally}, \mathsf{VerifyTally})$$

*Definition 2.4:* [(Perfect) Correctness] We require the following conditions (1) and (2) to hold.

1) Let $\mathsf{Abst}$ be a special symbol not in $\mathcal{M} \cup \{\perp\}$ that denotes that a voter did not cast her vote. (We need $\mathsf{Abst}$ to differentiate the case of a voter who did not cast a vote at all ($\mathsf{Abst}$) from the case of a voter who casts $\perp$ as her own vote but wishes to preserve the anonymity of her choice. However, in both cases, correctness guarantees that the result of the election equals the output of the tally function, and the input to the tally function is $\perp$ both when a voter casts $\perp$ and when a voter does not cast any vote.) For all $(\mathsf{Pk}, \mathsf{Sk}) \in \mathsf{Setup}(1^\lambda)$, all $m_1, \ldots, m_N \in \mathcal{M} \cup \{\perp, \mathsf{Abst}\}$, all $(\mathsf{Blt}_j)_{j=1}^N$ such that for all $j \in [N]$, $\mathsf{Blt}_j = \perp$ if $m_j = \mathsf{Abst}$, $\mathsf{Blt}_j \in \mathsf{Cast}(\mathsf{Pk}, j, m_j)$ if $m_j \in \mathcal{M}$ and $\mathsf{Blt}_j \in \mathsf{Cast}(\mathsf{Pk}, j, \perp)$ otherwise, the following two conditions (a) and (b) hold:
   a) For all $j \in [N]$, if $m_j \neq \mathsf{Abst}$ then $\mathsf{VerifyBallot}(\mathsf{Pk}, j, \mathsf{Blt}_j) = \mathsf{OK}$.
   b) if $(y, \gamma) \triangleq \mathsf{EvalTally}(\mathsf{Pk}, \mathsf{Sk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N)$ then it holds that:
   $y = F(m_1, \ldots, m_N)$ and $\mathsf{VerifyTally}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y, \gamma) = \mathsf{OK}$.
2) For all $(\mathsf{Pk}, \mathsf{Sk}) \in \mathsf{Setup}(1^\lambda)$, $\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N \in \{0,1\}^\star \cup \{\perp\}$, if $S \triangleq \{j |\ \mathsf{Blt}_j \neq \perp \wedge \mathsf{VerifyBallot}(\mathsf{Pk}, j, \mathsf{Blt}_j) = \perp\}$ and $\mathsf{Blt}'_1, \ldots, \mathsf{Blt}'_N$ are such that for all $j \in [N]$, $\mathsf{Blt}'_j = \mathsf{Blt}_j$ if $j \notin S$ and $\mathsf{Blt}'_j = \perp$ otherwise, it holds that:
   If $(y, \gamma) \triangleq \mathsf{EvalTally}(\mathsf{Pk}, \mathsf{Sk}, \mathsf{Blt}'_1, \ldots, \mathsf{Blt}'_N)$ then $\mathsf{VerifyTally}(\mathsf{Pk}, \mathsf{Blt}'_1, \ldots, \mathsf{Blt}'_N, y, \gamma) = \mathsf{OK}$.

Fig. 1. Definition of correctness

*Definition 2.5:* [Verifiability] We require the following conditions (1) and (2) to hold.

1) For all $\mathsf{Pk} \in \{0,1\}^\star, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N \in \{0,1\}^\star \cup \{\perp\}$, there exist unique $m_1, \ldots, m_N \in \mathcal{M} \cup \{\perp\}$ such that for all $y \in \{0,1\}^\star \cup \{\perp\}$ and $\gamma$ in $\{0,1\}^\star$, if $S \triangleq \{j |\ \mathsf{Blt}_j \neq \perp \wedge \mathsf{VerifyBallot}(\mathsf{Pk}, j, \mathsf{Blt}_j) = \perp\}$ and $\mathsf{Blt}'_1, \ldots, \mathsf{Blt}'_N$ are such that for all $j \in [N]$, $\mathsf{Blt}'_j = \mathsf{Blt}_j$ if $j \notin S$ and $\mathsf{Blt}'_j = \perp$ otherwise, it holds that:
   if $\mathsf{VerifyTally}(\mathsf{Pk}, \mathsf{Blt}'_1, \ldots, \mathsf{Blt}'_N, y, \gamma) = \mathsf{OK}$ then $y = F(m_1, \ldots, m_N)$.
2) For all $\mathsf{Pk} \in \{0,1\}^\star$, all $k \in [N]$, $i_1, \ldots, i_k \in [N]$, all $m_{i_1}, \ldots, m_{i_k} \in \mathcal{M} \cup \{\perp\}$, all $\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N \in \{0,1\}^\star \cup \{\perp\}$ such that for all $j \in [k]$, $\mathsf{Blt}_j \in \mathsf{Cast}(\mathsf{Pk}, j, m_{i_j})$ and $\mathsf{VerifyBallot}(\mathsf{Pk}, j, \mathsf{Blt}_j) = \mathsf{OK}$, if $S \triangleq \{j |\ \mathsf{Blt}_j \neq \perp \wedge \mathsf{VerifyBallot}(\mathsf{Pk}, j, \mathsf{Blt}_j) = \perp\}$ and $\mathsf{Blt}'_1, \ldots, \mathsf{Blt}'_N$ are such that for all $j \in [N]$, $\mathsf{Blt}'_j = \mathsf{Blt}_j$ if $j \notin S$ and $\mathsf{Blt}'_j = \perp$ otherwise, it holds that:
   if there exist $y \in \{0,1\}^\star \cup \{\perp\}$ and $\gamma \in \{0,1\}^\star$ such that $\mathsf{VerifyTally}(\mathsf{Pk}, \mathsf{Blt}'_1, \ldots, \mathsf{Blt}'_N, y, \gamma) = \mathsf{OK}$, then $y$ is compatible with $m_{i_1}, \ldots, m_{i_k}$ at indices $i_1, \ldots, i_k$.

Fig. 2. Definition of verifiability

$$\mathsf{Priv}_{\mathcal{A}}^{N,\mathcal{M},\Sigma,F,\mathsf{EVOTE}}(1^\lambda)$$

- Setup phase. $\mathcal{C}$ generates $(\mathsf{Pk}, \mathsf{Sk}) \leftarrow \mathsf{Setup}(1^\lambda)$, chooses a random bit $b \leftarrow \{0,1\}$ and runs $\mathcal{A}$ on input $\mathsf{Pk}$.
- Query phase. $\mathcal{A}$ outputs two tuples $M_0 \triangleq (m_{0,1}, \ldots, m_{0,N})$ and $M_1 \triangleq (m_{1,1}, \ldots, m_{1,N})$, and a set $S \subset [N]$. (The set $S$ contains the indices of the strings in the tuples that are possibly dishonest ballots. The strings in the tuples whose indices are not in $S$ are supposed to be votes to be given as input to the $\mathsf{Cast}$ algorithm.)
- Challenge phase. The challenger does the following. For all $j \in [N]$, if $j \in S$, set $\mathsf{Blt}_j \triangleq m_{b,j}$, else set $\mathsf{Blt}_j \leftarrow \mathsf{Cast}(\mathsf{Pk}, j, m_{b,j})$. For all $j \in S$, if $\mathsf{VerifyBallot}(\mathsf{Pk}, j, \mathsf{Blt}_j) = \perp$, set $\mathsf{Blt}_j \triangleq \perp$. Compute $(y, \gamma) \leftarrow \mathsf{EvalTally}(\mathsf{Pk}, \mathsf{Sk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N)$ and return $(\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y, \gamma)$ to the adversary.
- Output. At some point the adversary outputs its guess $b'$.
- Winning condition. The adversary wins the game if all the following conditions hold:
  1) $b' = b$.
  2) For all $j \in S, m_{0,j} = m_{1,j}$. (That is, if the adversary submits a dishonest ballot, it has to be the same in both tuples.)
  3) For all $d_1, \ldots, d_N \in \mathcal{M} \cup \{\perp\}$, for all $j \in [N]$, let $m'_{0,j} \triangleq m'_{1,j} \triangleq d_j$ if $j \in S$, and for all $b \in \{0,1\}$ let $m'_{b,j} \triangleq m_{b,j}$ if $m_{b,j} \in \mathcal{M}$ and $m'_{b,j} \triangleq \perp$ if $m_{b,j} \notin \mathcal{M}$. Then, $F(m'_{0,1}, \ldots, m'_{0,N}) = F(m'_{1,1}, \ldots, m'_{1,N})$.
  (That is, the tally function outputs the same result on input both tuples, even if the ballots corresponding to indices in $S$ are replaced by arbitrary messages in $\mathcal{M} \cup \{\perp\}$.)

Fig. 3. Definition of privacy

is formalized by means of the game $\mathsf{Priv}_{\mathcal{A}}^{N,\mathcal{M},\Sigma,F,\mathsf{EVOTE}}$ between a stateful adversary $\mathcal{A}$ and a *challenger* $\mathcal{C}$. We describe the game in Fig. 3.

In the game $\mathsf{Priv}_{\mathcal{A}}^{N,\mathcal{M},\Sigma,F,\mathsf{EVOTE}}$, the advantage of adversary $\mathcal{A}$ is defined as

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{EVOTE},\mathsf{Priv}}(1^\lambda) \triangleq |\Pr[\mathsf{Priv}_{\mathcal{A}}^{N,\mathcal{M},\Sigma,F,\mathsf{EVOTE}}(1^\lambda) = 1] - 1/2|$$

*Definition 2.6:* [Privacy] An EVOTE for parameters $(N, \mathcal{M}, \Sigma, F)$ is *private* or IND-Secure if the advantage of all non-uniform PPT adversaries $\mathcal{A}$ is at most negligible in $\lambda$ in the game $\mathsf{Priv}_{\mathcal{A}}^{N,\mathcal{M},\Sigma,F,\mathsf{EVOTE}}$.

The privacy definitions that we use here are simple and do not capture vote replay attacks, see e.g. [55]. Such attacks are easily prevented by enforcing ballot independence. This can e.g. be done by appending a proof of knowledge of the plaintext in the ballots of the voters. Presently, this has not been done in the NIWI setting, so we will disregard this point for clarity. However, we stress that it is easy to change the schemes to satisfy full privacy definitions within the framework of having trust for privacy, but not for verifiability.

Our privacy definition is inspired by the one of Benaloh [56], also called "PRIV" in [54], which we reformulate by using modern terminology and we modify to withstand the attacks shown in [54] by adding the third winning condition. One might think that an indistinguishability-based definition like ours is too weak. The reason why we introduce our definition and we do not use the definition in [54] is the following. The definition in [54] requires that a real proof must be indistinguishable from a simulated proof, and thus cannot be fulfilled by a construction based on witness-indistinguishable proofs. Luckily, in Appendix A, we show that our construction fulfills a simulation-based privacy definition like the one in [54] when the tally function is invertible, which includes functions of interest in e-voting.

### F. Remarks on Our Definitions

Our definitions assume that algorithm VerifyBallot is run on input each ballot before running algorithm VerifyTally. The ballots that are input to VerifyTally are replaced by $\perp$ if they were not accepted by VerifyBallot. Another possibility would be to let VerifyTally do this task itself.

We require that VerifyBallot and VerifyTally be deterministic algorithms. Alternatively, they can be defined as PPT, but then the definition of verifiability would have to be changed accordingly to hold with probability 1 over the random coins of the algorithms.

Our definition is parameterized by the number of voters $N$. It is possible to define a more restricted eVote that may possibly be "unbounded". Note that our definition is more general and, for instance, takes into account e-voting schemes in which the public key is of size proportional to the number of voters.

## III. Building Blocks

Our construction uses IND-CPA public-key encryption with perfect correctness and unique secret key, perfectly binding commitment schemes, and (one-message) non-interactive witness-indistinguishable proof systems with perfect soundness for NP [28] (see also [29], [29]–[31], [57]). In this section, we describe briefly those primitives. We give a more complete description in Appendix B.

An IND-CPA secure PKE scheme consists of three PPT algorithms $(\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt})$. Algorithm Setup, on input $1^\lambda$, outputs public key Pk and decryption key Sk. Algorithm Encrypt, on input message $m$ and public key Pk, outputs ciphertext Ct. Algorithm Decrypt, on input ciphertext Ct and decryption key Sk, outputs $m$. We require a PKE scheme that is IND-CPA secure and fulfills the perfect correctness and unique secret key properties. PKE schemes with those properties are known in the literature [41], [58] and can be constructed, e.g., from the Decision Linear assumption [41].

A commitment scheme Com is a PPT algorithm that takes as input a string $x$ and randomness $r \in \{0,1\}^k$ and outputs $\mathsf{com} \leftarrow \mathsf{Com}(x; r)$. We use a commitment scheme that is perfectly binding and computationally hiding. A perfectly binding non-interactive commitment scheme can be constructed from one-way permutations.

A non-interactive proof system for a language $L$ with a PPT relation $R$ is a tuple of algorithms $(\mathsf{Prove}, \mathsf{Verify})$. Prove receives as input a statement $x$ and a witness $w$ and outputs a proof $\pi$. Verify receives as input a statement $x$ and a proof $\pi$ and outputs a symbol in $\{\mathsf{OK}, \perp\}$. We use a (one-message) non-interactive proof system that fulfills the perfect completeness, perfect soundness and witness-indistinguishability properties. We refer to it as a (one-message) NIWI proof system.

## IV. Our eVote

Let $N$ be the number of voters and let $F$ be a tally function with message space $\mathcal{M}$. We present an eVote scheme EVOTE that is IND-Secure and verifiable.

*Definition 4.1:* [EVOTE] Let $\mathcal{E} = (\mathcal{E}.\mathsf{Setup}, \mathcal{E}.\mathsf{Encrypt}, \mathcal{E}.\mathsf{Decrypt})$ be a public-key encryption scheme with perfect correctness and unique secret key. Let Com be a perfectly binding commitment scheme. Let $\mathsf{NIWI}^{\mathsf{enc}} = (\mathsf{Prove}^{\mathsf{enc}}, \mathsf{Verify}^{\mathsf{enc}})$ and $\mathsf{NIWI}^{\mathsf{dec}} = (\mathsf{Prove}^{\mathsf{dec}}, \mathsf{Verify}^{\mathsf{dec}})$ be two NIWI proof systems for the relations $\mathsf{R}^{\mathsf{enc}}$ and $\mathsf{R}^{\mathsf{dec}}$, which we specify in Fig. 5 and Fig. 6. We define in Fig. 4 an $(N, \mathcal{M}, \Sigma, F)$-eVote

$$\mathsf{EVOTE}^{N,\mathcal{M},\Sigma,F,\mathcal{E},\mathsf{Com},\mathsf{NIWI}^{\mathsf{enc}},\mathsf{NIWI}^{\mathsf{dec}}} = (\mathsf{Setup}, \mathsf{Cast}, \mathsf{VerifyBallot}, \mathsf{EvalTally}, \mathsf{VerifyTally})$$

Our eVote uses 3 instances of a public-key encryption (PKE) scheme in parallel. We need 3 instances to "engineer" multiple witnesses for the NIWI proof system, as explained below. To compute the tally, 2 instances are used. We require that the PKE scheme fulfills two properties: perfect correctness and unique secret key. In addition to the three public keys of the PKE scheme, the public key of the authority contains a perfectly binding commitment $Z$ to the bit 1, i.e., the public key is $\mathsf{Pk} = (\mathsf{Pk}_1, \mathsf{Pk}_2, \mathsf{Pk}_3, Z)$, where $Z = \mathsf{Com}(1)$. The commitment $Z$ is used to enable a trapdoor mode for $\mathsf{R}^{\mathsf{enc}}$. The secret key consists of the 3 corresponding secret keys $(\mathsf{Sk}_1, \mathsf{Sk}_2, \mathsf{Sk}_3)$ of the PKE.

- $\mathsf{Setup}(1^\lambda)$: on input the security parameter in unary, do the following.
  1) Choose randomness $r \leftarrow \{0,1\}^\lambda$ and set $Z = \mathsf{Com}(1; r)$.
  2) For all $l \in [3]$, choose randomness $s_l \leftarrow \{0,1\}^\lambda$ and run $(\mathcal{E}.\mathsf{Pk}_l, \mathcal{E}.\mathsf{Sk}_l) = \mathcal{E}.\mathsf{Setup}(1^\lambda; s_l)$.
  3) Output $\mathsf{Pk} = (\mathcal{E}.\mathsf{Pk}_1, \mathcal{E}.\mathsf{Pk}_2, \mathcal{E}.\mathsf{Pk}_3, Z)$ and $\mathsf{Sk} = (\mathcal{E}.\mathsf{Sk}_1, \mathcal{E}.\mathsf{Sk}_2, s_1, s_2, r)$. (As the randomness for the setup of our PKE scheme uniquely determines the secret key, it would be sufficient to just include the $s_l$'s in $\mathsf{Sk}$.)
- $\mathsf{Cast}(\mathsf{Pk}, j, v)$: on input the public key $\mathsf{Pk}$, the voter index $j \in [N]$, and a vote $v$, do the following.
  1) For all $l \in [3]$, choose randomness $r_l \leftarrow \{0,1\}^\lambda$ and compute $\mathsf{Ct}_{j,l} = \mathcal{E}.\mathsf{Encrypt}(\mathcal{E}.\mathsf{Pk}_l, v; r_l)$.
  2) Consider the relation $\mathsf{R}^{\mathsf{enc}}$ in Fig. 5. Run $\mathsf{Prove}^{\mathsf{enc}}$ on input the statement $(j, \mathsf{Ct}_1, \mathsf{Ct}_2, \mathsf{Ct}_3, \mathcal{E}.\mathsf{Pk}_1, \mathcal{E}.\mathsf{Pk}_2, \mathcal{E}.\mathsf{Pk}_3, Z)$ and the witness $(v, r_1, r_2, r_3)$ to compute a proof $\pi_j$. Output $\mathsf{Blt}_j = (\mathsf{Ct}_{j,1}, \mathsf{Ct}_{j,2}, \mathsf{Ct}_{j,3}, \pi_j)$.
- $\mathsf{VerifyBallot}(\mathsf{Pk}, j, \mathsf{Blt})$: on input the public key $\mathsf{Pk}$, the voter index $j \in [N]$, and a ballot $\mathsf{Blt}$, output $\mathsf{Verify}^{\mathsf{enc}}((j, \mathsf{Ct}_1, \mathsf{Ct}_2, \mathsf{Ct}_3, \mathcal{E}.\mathsf{Pk}_1, \mathcal{E}.\mathsf{Pk}_2, \mathcal{E}.\mathsf{Pk}_3, Z), \pi)$.
- $\mathsf{EvalTally}(\mathsf{Pk}, \mathsf{Sk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N)$: on input the public key $\mathsf{Pk}$, the secret key $\mathsf{Sk}$, and $N$ strings $(\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N)$ that can be either ballots cast by a voter or the special symbol $\perp$, do the following.
  1) For all $j \in [N]$, if $\mathsf{VerifyBallot}(\mathsf{Pk}, j, \mathsf{Blt}_j) = \perp$, set $\mathsf{Blt}_j = \perp$. If, for all $j \in [N]$, $\mathsf{Blt}_j = \perp$, then output $(y = \perp, \gamma = \perp)$.
  2) Else, for all $j \in [N], l \in [2]$,
$$m_{j,l} = \begin{cases} \perp & \text{if } \mathsf{Blt}_j = \perp, \\ \perp & \text{if } \mathsf{Blt}_j \neq \perp \wedge \mathcal{E}.\mathsf{Decrypt}(\mathsf{Ct}_{j,l}, \mathcal{E}.\mathsf{Sk}_l) \notin \mathcal{M}, \\ \mathcal{E}.\mathsf{Decrypt}(\mathsf{Ct}_{j,l}, \mathcal{E}.\mathsf{Sk}_l) & \text{otherwise.} \end{cases}$$
  3) For all $l \in [2]$, compute $y_l = F(m_{1,l}, \ldots, m_{N,l})$.
  4) If $y_1 = y_2$ then set $y = y_1$, else set $y = \perp$. (Here, in an honest execution, in which the ballots computed by the voters are replaced by $\perp$ if they are not accepted by the verification ballot algorithm, the "else" case will never happen.)
  5) Consider the relation $\mathsf{R}^{\mathsf{dec}}$ in Fig. 6. (If the indices $(i_1, i_2)$ in the witness of the relation $\mathsf{R}^{\mathsf{dec}}$ fulfill $i_1 = 1$ and $i_2 = 2$ (resp. $i_1 \neq 1$ or $i_2 \neq 2$), the statement or the proof is in real mode (resp. trapdoor mode).) Run $\mathsf{Prove}^{\mathsf{dec}}$ on input the statement $(\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, \mathcal{E}.\mathsf{Pk}_1, \mathcal{E}.\mathsf{Pk}_2, \mathcal{E}.\mathsf{Pk}_3, y)$ and the witness $(\mathcal{E}.\mathsf{Sk}_1, \mathcal{E}.\mathsf{Sk}_2, s_1, s_2, i_1 = 1, i_2 = 2)$ to compute a proof $\gamma$.
  6) Output $(y, \gamma)$.
- $\mathsf{VerifyTally}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y, \gamma)$: on input the public key $\mathsf{Pk}$, $N$ strings that can be either ballots cast by a voter or the special symbol $\perp$, a tally $y$ and a proof $\gamma$ of tally correctness, do the following. Replace $\mathsf{Blt}_j$'s with $\perp$ when $\mathsf{VerifyBallot}(\mathsf{Pk}, j, \mathsf{Blt}_j) = \perp$. Then, if $y = \perp$ and all $\mathsf{Blt}_j$'s are equal to $\perp$, output $\mathsf{OK}$. If $y = \perp$ but not all $\mathsf{Blt}_j$'s are equal to $\perp$, output $\perp$. Otherwise output the decision of $\mathsf{Verify}^{\mathsf{dec}}((\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, \mathcal{E}.\mathsf{Pk}_1, \mathcal{E}.\mathsf{Pk}_2, \mathcal{E}.\mathsf{Pk}_3, y), \gamma)$. Precisely, the algorithm does the following:
  1) For all $j \in [N]$, if $\mathsf{VerifyBallot}(\mathsf{Pk}, j, \mathsf{Blt}_j) = \perp$, set $\mathsf{Blt}_j = \perp$.
  2) If $y \neq \perp$, then output $\mathsf{Verify}^{\mathsf{dec}}((\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, \mathcal{E}.\mathsf{Pk}_1, \mathcal{E}.\mathsf{Pk}_2, \mathcal{E}.\mathsf{Pk}_3, y), \gamma)$.
  3) If $y = \perp$, then, if for all $j \in [N], \mathsf{Blt}_j = \perp$, output $\mathsf{OK}$, else output $\perp$.

Fig. 4. Our eVote $\mathsf{EVOTE}^{N,\mathcal{M},\Sigma,F,\mathcal{E},\mathsf{Com},\mathsf{NIWI}^{\mathsf{enc}},\mathsf{NIWI}^{\mathsf{dec}}}$

Our cast algorithm takes as input the public key $(\mathsf{Pk}_1, \mathsf{Pk}_2, \mathsf{Pk}_3, Z)$, the index $j$ of the voter[5] (for $j \in [N]$), and a vote $v$. The cast algorithm outputs a ballot $\mathsf{Blt}_j = (\mathsf{Ct}_{j,1}, \mathsf{Ct}_{j,2}, \mathsf{Ct}_{j,3}, \pi_j)$ for the $j$-th voter. A ballot consists of three ciphertexts and of a proof that either the three ciphertexts encrypt the same message in the message space $\mathcal{M} \cup \{\perp\}$ (real mode) or $Z$ is a commitment to 0 (trapdoor mode). Formally, the ballot contains a NIWI proof for the relation $\mathsf{R}^{\mathsf{enc}}$ in Fig. 5. The ballot verification algorithm runs the verification algorithm for the NIWI proof system for the relation $\mathsf{R}^{\mathsf{enc}}$.

The tally evaluation algorithm works as follows. For all $j \in [N]$, set a ballot to $\perp$ if the ballot does not pass the ballot verification algorithm. Else, for all $l \in [2]$, decrypt $\mathsf{Ct}_{j,l}$ with $\mathsf{Sk}_l$ to get $m_{j,l}$. Then, for all $l \in [2]$ compute $y_l = F(m_{1,l}, \ldots, m_{N,l})$, where for indices $j$ such that either $m_{j,l} \notin \mathcal{M}$ or the ballot is $\perp$, we set $m_{j,l} = \perp$. If the two $y_l$'s are equal to the *same* string $y$ then return this as the tally, otherwise return an error $y = \perp$. Finally, compute a

[5]The index is needed to associate a ballot with a unique voter. For instance, an eVote could require that each voter encrypts her ballot with a different PKE public key, adding a proof of well-formedness. The public key of the eVote would contain $N$ PKE's public keys, one for each voter, and so the statement of the proof would have to contain the index of the voter in the set $N$.

Relation $\mathsf{R}^{\mathsf{enc}}(x, w)$:

Instance: $x \stackrel{\triangle}{=} (j, \mathsf{Ct}_1, \mathsf{Ct}_2, \mathsf{Ct}_3, \mathcal{E}.\mathsf{Pk}_1, \mathcal{E}.\mathsf{Pk}_2, \mathcal{E}.\mathsf{Pk}_3, Z)$.

Witness: $w \stackrel{\triangle}{=} (m, r_1, r_2, r_3, u)$, where the $r_l$'s are the randomness used to compute the ciphertexts $\mathsf{Ct}_l$'s and $u$ is the randomness used to compute the commitment $Z$.

$\mathsf{R}^{\mathsf{enc}}(x, w) = 1$ if and only if either of the following two conditions hold:

1) **Real mode.** All 3 ciphertexts $(\mathsf{Ct}_1, \mathsf{Ct}_2, \mathsf{Ct}_3)$ encrypt the same string in $\mathcal{M} \cup \{\perp\}$.
   Precisely, for all $l \in [3]$, $\mathsf{Ct}_l = \mathcal{E}.\mathsf{Encrypt}(\mathcal{E}.\mathsf{Pk}_l, m; r_l)$ and $m \in \mathcal{M} \cup \{\perp\}$.

OR

2) **Trapdoor mode.** $Z$ is a commitment to 0.
   Precisely, $Z = \mathsf{Com}(0; u)$.

Fig. 5. Relation $\mathsf{R}^{\mathsf{enc}}$

NIWI proof $\gamma$ of the fact that $x = (\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, \mathsf{Pk}_1, \mathsf{Pk}_2, \mathsf{Pk}_3, y)$ satisfies the relation $\mathsf{R}^{\mathsf{dec}}$ in Fig. 6 using as witness $(\mathsf{Sk}_1, \mathsf{Sk}_2, s_1, s_2)$. Another part of the witness is the two indices $i_1, i_2 \in [3]$, $i_1 < i_2$, which determine the two columns of ciphertexts that are used to compute the tally. In the real mode described above, we have $i_1 = 1, i_2 = 2$, but we can also have trapdoor modes with other index choices, which is essential to prove privacy.

More in detail, to be able to prove that our eVote fulfills the privacy property, we need to show that the ballots that encrypt messages $m_{0,j}$ are indistinguishable from the ballots that encrypt messages $m_{1,j}$. To prove that, we need to be able to switch the messages encrypted in the ballots from $m_{0,j}$ to $m_{1,j}$ and prove indistinguishability based on the IND-CPA property of the PKE scheme. If the ballot consisted of only one ciphertext, we would not be able to that, because the witness-indistinguishability property of the proof system for $\mathsf{R}^{\mathsf{dec}}$ would not suffice. To solve this problem, we need to "engineer" multiple witnesses for the NIWI.

Our solution consists in using ballots with three ciphertexts, and a tally evaluation algorithm that decrypts two of them. In real mode, $\mathsf{R}^{\mathsf{enc}}$ proves that the three ciphertexts encrypt the same message, and $\mathsf{R}^{\mathsf{dec}}$ uses the first two ciphertexts to prove correctness of the tally. In the security prove, we use the trapdoor mode of $\mathsf{R}^{\mathsf{enc}}$ and $\mathsf{R}^{\mathsf{dec}}$. The trapdoor mode of $\mathsf{R}^{\mathsf{enc}}$ allows us to switch the message encrypted in one the ciphertexts from $m_{0,j}$ to $m_{1,j}$. To prove indistinguishability based on the IND-CPA property of the PKE scheme, we switch the ciphertext that is not used in $\mathsf{R}^{\mathsf{dec}}$ as follows. When $i_1 = 1$ and $i_2 = 2$ in $\mathsf{R}^{\mathsf{dec}}$, and starting with a ballot the encrypts $m_{0,j}$, we switch the message encrypted in the third ciphertext. Then we modify the witness of $\mathsf{R}^{\mathsf{dec}}$ to $i_1 = 1$ and $i_2 = 3$ and prove indistinguishability by using the witness-indistinguishability property of the NIWI proof system. This allows us to switch now the message encrypted in the second ciphertext. By repeating these steps, we obtain ballots that encrypt $m_{1,j}$.

The tally verification algorithm verifies $(y, \gamma)$ by using the verification algorithm of the NIWI system. Additionally, if either (1) not all ballots are $\perp$ and $y = \perp$, or (2) all ballots are $\perp$ and $y \neq \perp$, the tally verification algorithm outputs $\perp$.

The reason for the latter is the following. First, note that, in our scheme, the ballots that are rejected by the ballot verification algorithm are replaced by $\perp$ as input to the tally evaluation algorithm. Our tally functions must fulfill a very natural property: $F(m_1, \ldots, m_N) = \perp$ iff $m_1 = \perp, \ldots, m_N = \perp$ (cf. Def. 2.2). That is, if at least one message is valid, then it has to be "counted". We prove that, if the public key is honestly generated, the tally evaluation algorithm never returns $\perp$ on input a tuple of possibly dishonest ballots. Therefore, except for the case that all the ballots are invalid, a tally $y = \perp$ may only occur if the authority acted dishonestly and, consequently, the tally verification algorithm should not accept $y = \perp$.

### A. *Verifiability of our eVote*

We describe why our eVote fulfills the verifiability property. A detailed proof is given in Appendix C. This property consists of two conditions defined in Def. 2.5.

First, we show that our scheme fulfills the first condition. The first condition states that, if each ballot and the proof of correctness of the tally issued by the authority are verified successfully by the respective algorithms, then each ballot (possibly computed on input a maliciously generated public key) must be associated with a unique message $m_i \in \mathcal{M} \cup \{\perp\}$, and the result $y$ claimed by the authority equals $F(m_1, \ldots, m_n)$.

We show that our construction fulfills the first condition as follows. Our tally verification algorithm only accepts a tally $y = \perp$ when all the ballots are invalid. Therefore, (1) the authority is not able to wrongly claim that a tally is $\perp$. Furthermore, (2) the authority cannot output two tallies $y_0, y_1$

Relation $\mathsf{R}^{\mathsf{dec}}(x, w)$:

Instance: $x \triangleq (\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, \mathcal{E}.\mathsf{Pk}_1, \mathcal{E}.\mathsf{Pk}_2, \mathcal{E}.\mathsf{Pk}_3, y)$. (Recall that a ballot is set to $\perp$ if either the corresponding voter did not cast her vote or her ballot is not accepted by the ballot verification algorithm.)

Witness: $w \triangleq (\mathcal{E}.\mathsf{Sk}'_1, \mathcal{E}.\mathsf{Sk}'_2, s_1, s_2, i_1, i_2)$, where the $s_l$'s are the randomness used to generate the secret keys and public keys (which are known to the authority who set up the system).

$\mathsf{R}^{\mathsf{dec}}(x, w) = 1$ if and only if the following conditions hold: $i_1 \neq i_2$; 2 of the secret keys corresponding to indices $\mathcal{E}.\mathsf{Pk}_{i_1}, \mathcal{E}.\mathsf{Pk}_{i_2}$ are constructed using honestly generated public and secret key pairs and are equal to $\mathcal{E}.\mathsf{Sk}'_1, \mathcal{E}.\mathsf{Sk}'_2$; and either $y = \perp$ or for all $l \in [2]$, $y = F(m^l_1, \ldots, m^l_N)$ and for all $j \in [N]$, if $\mathsf{Blt}_j \neq \perp$ then for $l \in [2]$, $\mathcal{E}.\mathsf{Sk}_{i_l}$ decrypts ciphertext $\mathsf{Ct}_{j,i_l}$ in $\mathsf{Blt}_j$ to $m^{i_l}_j \in \mathcal{M}$; and for all $l \in [2]$, $m^l_j = \perp$ if either $\mathsf{Blt}_j = \perp$ or $\mathcal{E}.\mathsf{Sk}_{i_l}$ decrypts $\mathsf{Ct}_{j,i_l}$ to a string $\notin \mathcal{M}$.

Precisely, $\mathsf{R}^{\mathsf{dec}}(x, w) = 1$ if and only if the following conditions hold. In the following, items (1) and (3) are not actually conditions that have to be checked but are steps needed to define (note the use of "$\triangleq$") the variables $\mathcal{E}.\mathsf{Pk}_{i_l}$'s, $\mathcal{E}.\mathsf{Sk}_{i_l}$'s and $m^{i_l}_j$'s that are used in the checks (2) and (4).

1) For all $l \in [2], (\mathcal{E}.\mathsf{Pk}_{i_l}, \mathcal{E}.\mathsf{Sk}_{i_l}) \triangleq \mathcal{E}.\mathsf{Setup}(1^\lambda; s_l)$.
2) For all $l \in [2], \mathcal{E}.\mathsf{Sk}'_l = \mathcal{E}.\mathsf{Sk}_{i_l}$.
3) For all $j \in [N], l \in [2]$,

$$m^{i_l}_j \triangleq \begin{cases} \perp & \text{if } \mathsf{Blt}_j = \perp, \\ \perp & \text{if } \mathsf{Blt}_j \neq \perp \wedge \mathcal{E}.\mathsf{Decrypt}(\mathsf{Ct}_{j,i_l}, \mathcal{E}.\mathsf{Sk}_{i_l}) \notin \mathcal{M}, \\ \mathcal{E}.\mathsf{Decrypt}(\mathsf{Ct}_{j,i_l}, \mathcal{E}.\mathsf{Sk}_{i_l}) & \text{otherwise.} \end{cases}$$

4) $(y = \perp) \vee$ (for all $l \in [2]$, $y = F(m^{i_l}_1, \ldots, m^{i_l}_N)$).
5) $i_1 \neq i_2$.

(Note that $\mathcal{E}.\mathsf{Sk}'_1$ and $\mathcal{E}.\mathsf{Sk}'_2$ do not necessarily have to correspond to the first two secret keys. If the indices $(i_1, i_2)$ fulfill $i_1 = 1$ and $i_2 = 2$ (resp. $i_1 \neq 1$ or $i_2 \neq 2$), the statement or the proof is in real mode (resp. trapdoor mode).)

Fig. 6. Relation $\mathsf{R}^{\mathsf{dec}}$

such that $y_0 \neq y_1, y_0, y_1 \neq \perp$ along with proofs $\gamma_0$ and $\gamma_1$ that are accepted by the tally verification algorithm. We use a contradiction to show (2). Let us assume that there exist two results $y_0, y_1 \neq \perp$ such that $y_0 \neq y_1$, and two proofs $\gamma_0, \gamma_1$ that are accepted by the tally verification algorithm. By the unique secret key property, the decryption of the ciphertexts in the ballots produces a unique result. By the pigeon principle, there exists one index $i^\star \in [3]$ used by both proofs. Therefore, it must be the case that either $y_0 = y_1 = \perp$ or $y_0$ and $y_1$ are equal to the evaluation of the tally function $F$ on input the messages obtained by decrypting the ciphertexts. Consequently, $y_0, y_1 \neq \perp$ such that $y_0 \neq y_1$ is a contradiction. Therefore, (1) and (2) together imply that the result $y$ claimed by the authority equals $F(m_1, \ldots, m_n)$.

The second condition requires that, even when the adversary generates the public key, if honest voters cast a ballot that is accepted by the ballot verification algorithm, then the ballot has to be "counted". We show that the second condition also holds. First, we note that the authority can only create a dishonest public key by setting the commitment dishonestly. The reason is that the authority has to prove that the public key of the PKE scheme is honestly generated, i.e., the perfect soundness of the NIWI ensures that the public key for the PKE scheme is honestly generated. The perfect correctness of the PKE scheme ensures that a ballot that encrypts $m$ will be decrypted to $m$. Therefore, the NIWI and the PKE scheme guarantee that an honestly computed ballot[6] for the $j$-th voter that encrypts message $m$ will always be "counted", i.e., for any $(y, \gamma)$ pair that is accepted by the tally verification algorithm, $y$ will be compatible with $m$ at index $j$ according to Def. 2.1.

### B. On The Reusability of the Public Parameters

Our definition of verifiability does not prevent the following undesirable case. Consider an ill-formed ballot $\mathsf{Blt}_1$. Consider other valid ballots $\mathsf{Blt}_2, \ldots, \mathsf{Blt}_N$ that encrypt respectively $v_2, \ldots, v_N$. The authority is able to compute a tally

[6]Here, "honestly computed ballot" just means that it is computed by the voter using the Cast algorithm on input the public key of the authority, which could be honestly or dishonestly created. By design of our construction, an honestly generated ballot computed on input an honestly created public key has the same distribution of an honestly created ballot computed on input any possibly dishonest public key whenever the authority is able to compute proofs of tally correctness that are accepted by the tally verification algorithm.

$y = F(v_1, \ldots, v_N)$ and a valid proof of tally correctness. Consider now other valid ballots $\mathsf{Blt}'_2, \ldots, \mathsf{Blt}'_N$ that encrypt $v'_2, \ldots, v'_N$. The authority can possibly compute another tally $y' = F(v'_1, v'_2, \ldots, v'_N)$ and another proof of tally correctness. The problem is that the ill-formed ballot $\mathsf{Blt}_1$ can be decrypted to more than one message.

This does not contradict our definition because, for $\mathsf{Blt}_1$, $\ldots, \mathsf{Blt}_N$, there still exist messages $v_1, \ldots, v_N$ that satisfy the statement of the definition, i.e., given $\mathsf{Pk}$ and $\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N$, the authority cannot output two different results and two valid proofs of tally correctness for each of them. However, it can occur that for $\mathsf{Pk}, \mathsf{Blt}_1, \mathsf{Blt}'_2, \ldots, \mathsf{Blt}'_N$, there are different messages that satisfy the definition. We remark that the public key $\mathsf{Pk}$ does not change.

Let us present a concrete example. Consider two $0/1$ elections with only 2 voters. A ballot could possibly be *reused* in the second election, i.e., if the public parameters of the system are reused, the same ballot can be cast again. Given an ill-formed ballot $\mathsf{Blt}_1$, there could exist two ballots $\mathsf{Blt}_2$ and $\mathsf{Blt}'_2$ such that, in an election with ballots $\mathsf{Blt}_1$ and $\mathsf{Blt}_2$, the result is 2, and, in a election with ballots $\mathsf{Blt}_1$ and $\mathsf{Blt}'_2$, the result is 0. This can only happen if the first ballot is "associated" with vote 1 in the first election and with vote 0 in the second election. Therefore, the first and the second elections are incoherent. More undesirable issues would emerge if different tally functions could be computed in different elections carried out with the same parameters and ballots.

A stronger definition could state that, for all $\mathsf{Pk}$ and all $\mathsf{Blt}_1$, there exists $m_1$ such that, for all $\mathsf{Blt}_2, \ldots, \mathsf{Blt}_N$, there exist $m_2, \ldots, m_n$ such that the authority is only able to output a tally $y = F(m_1, \ldots, m_N)$ along with a valid proof of tally correctness. We note that this is a simplification because a general definition should take into account multiple dishonest voters.

Fortunately, in our e-voting model, as well as in other traditional models, the parameters cannot be reused through different elections. Therefore, the above-mentioned problem does not occur.

In a stronger model in which the parameters can be reused, our construction would not be secure. Nevertheless, the inconsistency of results through different elections would occur only in the case that a malicious authority sets the commitment in the public key dishonestly to 0, which allows the computation of ill-formed ballots.

This state of affairs could be paralleled to the case of garbled circuits, where the original one-time version [59], [60] can be based on the minimal assumption of existence of one-way functions, whereas the reusable variant [61] is known to be implementable only under stronger assumptions. Similarly, in functional encryption, the schemes with bounded security [62], [63] can be based just on public-key encryption, whereas the unbounded secure variants are only known to be implementable under very strong assumptions [64]. For instance, the scheme of Sahai and Seyalioglu [62] becomes completely insecure when the adversary can decrypt a ciphertext with two different secret keys, exactly as it occurs for our scheme.

### C. Privacy of our eVote

We show now that our scheme fulfills the privacy property defined in Def. 2.6. Here we summarize the proof. In Appendix D, we describe the proof in detail. We stress that, for privacy to hold, the authority must be honest and thus the public key is honestly generated.

In the security proof, we consider a sequence of hybrid experiments. First, we define an experiment $H^Z$ in which the commitment in the public key is a commitment to 0. We show that $H^Z$ is indistinguishable from the real experiment under the computationally hiding property of the commitment.

Second, we define an event $E^1$ in experiment $H^Z$. In $E^1$, the adversary submits a ballot that is accepted by the ballot verification algorithm but, when decrypting the three ciphertexts in the ballot, the three decrypted messages in $\mathcal{M} \cup \{\perp\}$ are not equal. We show that the probability of $E^1$ is negligible under the computationally hiding property of the commitment. More concretely, we show that if $E^1$ occurs with non-negligible probability, then the adversary can be used to distinguish a commitment to 0 from a commitment to 1. We note that if $Z$ is a commitment to 1, then the perfect soundness of the NIWI guarantees that the adversary can never submit an ill-formed ballot that is accepted by the ballot verification algorithm. Therefore, the probability $\bar{E}^1$ of its complement is overwhelming.

We recall that the adversary sends two tuples $V_0 = (m_{0,1}, \ldots, m_{0,N})$ and $V_1 = (m_{1,1}, \ldots, m_{1,N})$, and a set $S \subset [N]$ that contains the indices of the strings of arbitrary ballots. The hybrid experiments after $H^Z$ work as follows.

- Hybrid experiment $H_1$ is equal to the experiment $H^Z$, except that the challenger sets the bit $b = 0$.
- Hybrid experiment $H_2$ switches the message encrypted in the third ciphertext in any ballot to encrypt $m_{1,j}$ instead of $m_{0,j}$. More in detail, for $k = 0$ to $N$, we define a sequence of hybrid experiments $H_2^k$. $H_2^k$ is identical to $H_1$, except that, for all $j = 1, \ldots, k$ such that $j \notin S$, the challenger computes the third ciphertext of the ballot on input $m_{1,k}$. Therefore, $H_2^0$ is identical to $H_1$, while $H_2^N$ is identical to $H_2$. We show that $H_2^k$ and $H_2^{k+1}$ are indistinguishable thanks to the IND-CPA property of the PKE scheme.
- Hybrid experiment $H_3$ is identical to experiment $H_2$, except that the challenger computes the NIWI proof $\gamma$ on input a witness that contains indices $(1, 3)$ and secret keys $\mathsf{Sk}_1, \mathsf{Sk}_3$, instead of indices $(1, 2)$ and secret keys $\mathsf{Sk}_1$, $\mathsf{Sk}_2$. We show that $H_3$ and $H_2$ are indistinguishable thanks to the witness-indistinguishability property of the NIWI proof. Because $\bar{E}^1$ occurs with overwhelming probability, any ballot in $S$ is either replaced by $\perp$, if the ballot verification algorithm does not accept it, or decrypted to the same value in $H_2$ and $H_3$. Therefore, the tally evaluation algorithm outputs the same tally in $H_2$ and $H_3$.
- Hybrid experiment $H_4$ is identical to $H_3$, except that the second ciphertext in any ballot encrypts $m_{1,j}$ instead

TABLE I
HYBRID GAMES TO PROVE FULFILLMENT OF THE PRIVACY PROPERTY.

| Exp | $(\mathsf{Ct}_{j,1}, \mathsf{Ct}_{j,2}, \mathsf{Ct}_{j,3})$ | $\mathsf{Sk}$ index | $\gamma$ | Security |
|---|---|---|---|---|
| $H_1$ | $(m_{0,j}, m_{0,j}, m_{0,j})$ | (1,2,3) | R | - |
| $H_2$ | $(m_{0,j}, m_{0,j}, m_{1,j})$ | (1,2,3) | R | IND-CPA |
| $H_3$ | $(m_{0,j}, m_{0,j}, m_{1,j})$ | (1,2,3) | T | WI |
| $H_4$ | $(m_{0,j}, m_{1,j}, m_{1,j})$ | (1,2,3) | T | IND-CPA |
| $H_5$ | $(m_{0,j}, m_{1,j}, m_{1,j})$ | (1,2,3) | T | WI |
| $H_6$ | $(m_{1,j}, m_{1,j}, m_{1,j})$ | (1,2,3) | T | IND-CPA |
| $H_7$ | $(m_{1,j}, m_{1,j}, m_{1,j})$ | (1,2,3) | R | WI |

of $m_{0,j}$. More in detail, for $k = 0$ to $N$, we define a sequence of hybrid experiments $H_4^k$. $H_4^k$ is identical to $H_3$, except that, for all $j = 1, \ldots, k$ such that $j \notin S$, the challenger computes the second ciphertext of the ballot on input $m_{1,k}$. Therefore, $H_4^0$ is identical to $H_3$, while $H_4^N$ is identical to $H_4$. We show that $H_4^k$ and $H_4^{k+1}$ are indistinguishable thanks to the IND-CPA property of the PKE scheme.

The remaining hybrid experiments are symmetrical to the ones described above. In $H_5$, the witness used to compute the NIWI proof contains the indices $(2, 3)$ and secret keys $\mathsf{Sk}_2, \mathsf{Sk}_3$, and indistinguishability between $H_5$ and $H_4$ follows from the witness-indistinguishability property of the NIWI proof. In $H_6$, the first ciphertext of each ballot encrypts $m_{1,j}$ instead of $m_{0,j}$ and indistinguishability between $H_6$ and $H_5$ follows from the IND-CPA property of the PKE scheme. Finally, in $H_7$ the witness used to compute the NIWI proof contains the indices $(1, 2)$ and secret keys $\mathsf{Sk}_1, \mathsf{Sk}_2$, and indistinguishability between $H_7$ and $H_6$ follows from the witness-indistinguishability property of the NIWI proof.

The sequence of hybrid experiments after $H^Z$ is summarized in Table IV-C. In Table IV-C, the first column shows the name of the hybrid experiment. The second column shows the three messages that are encrypted in the 3 ciphertexts $\mathsf{Ct}_{j,1}, \mathsf{Ct}_{j,2}, \mathsf{Ct}_{j,3}$ contained in the challenge ballot $\mathsf{Blt}_j$ associated with voter $j$. The text in blue in the "$\mathsf{Sk}$ index" column denotes the indices used as witness in the proof $\gamma$. As mentioned above, if such blue indices correspond to the set $\{1, 2\}$ (resp. to a set different from $\{1, 2\}$) we say that the statement or proof is in real mode (resp. trapdoor mode), which we denote by $R$ (resp. $T$) in the column $\gamma$. The text in red indicates the difference from the previous hybrid experiment.

Thanks to the hybrid experiment $H^Z$, in the subsequent hybrid experiments we can show indistinguishability by using the IND-CPA property of the PKE scheme. The reason is that, thanks to $H^Z$, the NIWI proof in the ballots can be a proof that the commitment in the public key is a commitment to 0. Therefore, we avoid the computation of a proof that shows that the three ciphertexts encrypt the same message, which allows us to switch the message encrypted in one of the ciphertexts and prove indistinguishability by using the IND-CPA assumption.

When showing indistinguishability between the hybrid experiments after $H^Z$, we have to guarantee that, when we switch the indices used as witness for the NIWI proof of tally correctness, the tally does not change. To illustrate this issue, suppose that, in an adversarial ballot, the first two ciphertexts encrypt the same message $x$ but the third one encrypts a different message $z$. Then the tally computed by the secret keys for indices $\{1, 2\}$ could differ from the one computed with secret keys for indices $\{2, 3\}$. In that case, we cannot prove indistinguishability between a hybrid experiment where the NIWI witness comprises $\mathsf{Sk}_1, \mathsf{Sk}_2$ and a hybrid experiment where the NIWI witness comprises $\mathsf{Sk}_2, \mathsf{Sk}_3$. To solve this issue, we show that event $E^1$ occurs with negligible probability. Therefore, it is sufficient to analyze the advantage of the adversary in the hybrid experiments after $H^Z$ conditioned on the occurrence of $\bar{E}^1$ (i.e., the complement of $E^1$).

We would like to remark the subtle difference between ill-formed and invalid ballots. An ill-formed ballot is a ballot that is not in the range of the cast algorithm. However, an ill-formed ballot could be valid in the sense that, along with other (possibly ill- or well- formed) $N-1$ ballots, the authority obtains a tally, i.e., the tally obtained when decrypting the first and the second ciphertext in the ballots is the same. An ill-formed ballot can be computed when the commitment in the public key is computed dishonestly.

The event $\bar{E}^1$ may occur even if the adversary submits an ill-formed ballot that is accepted by the ballot verification algorithm. In fact, if a (non-honestly computed) ballot is formed by strings that are not in the ciphertext space of the encryption algorithm of the PKE, but decryption of those strings outputs the same message, such a ballot is not considered invalid.

Note also that the proof of well-formedness of the ballots states that the encrypted messages may be equal to $\perp$. Ballots that encrypt $\perp$ are blank ballots. We consider tally functions in which the symbol $\perp$ indicates a blank vote. For example, in case of an eVote for the sum function in which $\perp$ is counted as 0, an adversary should not be able to distinguish three ballots that encrypt $(1, 1, \perp)$ from three ballots that encrypt $(1, \perp, 1)$.

## V. eVOTE WITH MULTIPLE AUTHORITIES

We sketch how to generalize our eVote to fit a model with multiple authorities. In this model, the tally evaluation algorithm is run by a set of authorities. The privacy property must hold if not all the authorities are corrupt. Our generalized scheme guarantees a statistical verifiability property (see below), which assumes that there is at least one honest voter.

First, we note that the multi-string model of Groth and Ostrovsky [44] does not provide a solution to this problem. The multi-string model assumes that the majority of the parties that set up the CRSs are honest. It does not guarantee soundness, which would provide verifiability in our application, when *all* those parties, which would be the authorities in our application, are corrupt. In the multi-string model, there is a trade-off between soundness and zero-knowledge. Namely, soundness could hold when all the authorities are corrupt, but then zero-knowledge does not hold. Zero-knowledge is guaranteed only when there is a majority of honest authorities. In contrast, our generalized scheme fulfills the privacy property when at least one authority is honest.

### A. Sketch of the Construction

Our generalized construction works for tally functions that can be represented as polynomials. Such tally functions comprise many functions of interest for e-voting. For simplicity, henceforth we only consider the case of the sum function with a binary message space. The general case follows by using Lagrange's polynomial interpolation.

Consider the sum function over a set of integers $S^k$, which we specify later. Consider $m$ authorities. Each authority $k \in [m]$ publishes a public key that consists of the public key of our eVote and, in addition, a commitment $\mathsf{com}_k$ to a tuple of $N$ 0's.

A ballot $\mathsf{Blt}_j$ for the $j$-th voter consists of the ballots $(\mathsf{Blt}_{j,1}, \ldots, \mathsf{Blt}_{j,m})$. For a vote $v_j$, each voter computes $m$ shares $v_{j,1}, \ldots, v_{j,m}$ whose sum is $v_j$. (Later we describe how the shares are computed in order to preserve privacy.) The ballot $\mathsf{Blt}_{j,k}$ for the $k$-th authority is computed following the cast algorithm of our eVote on input the share $v_{j,k}$. In addition, the voter adds a NIWI proof that either (the real statement) for all $k \in [m]$, $\mathsf{Blt}_{j,k}$ encrypts a number in $S^k$ such that the sum of the encrypted numbers is in $\{0, 1\}$ (for simplicity, here we do not consider messages equal to $\perp$) OR (the trapdoor statement) for all $k \in [m]$, $\mathsf{com}_k$ is a commitment to a tuple $(z_1, \ldots, z_N)$ such that $z_j$ is equal to the tuple $(\mathsf{Blt}_{j,1}, \ldots, \mathsf{Blt}_{j,m})$.

For each $k \in [m]$, the $k$-th authority computes the tally $y_k$ as in our eVote. The proof of correctness of the tally is a proof for the following modified relation: either (the real statement) the witness satisfies the relation $\mathsf{R}^{\mathsf{dec}}$ of our eVote and $\mathsf{com}_k$ is a commitment to 0 OR (the trapdoor statement) $\mathsf{com}_k$ is a commitment to a tuple $(z_1, \ldots, z_N)$ such that $z_j = \mathsf{Blt}_j$ for all $j \in [N]$, where $\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N$ are the $N$ ballots published on the public bulletin board.

Finally, the tally is computed by summing the tallies $y_k$'s output by each of the authorities to obtain $y$. We give more details below.

To support functions represented as polynomials, the following modifications should be applied. To compute the shares $v_{j,1}, \ldots, v_{j,m}$, the voter $j$ chooses a polynomial $p_j$ of degree $m-1$ such that $p_j(0)$ equals her vote $v_j$. The shares are the evaluation of $p_j$ on input $1, \ldots, m$. The tally is computed by using Lagrange interpolation.

### B. Verifiability of the Construction

We analyze now the verifiability of our generalized construction. If the commitments in the public key are computed honestly, we can show that the generalized construction fulfills the verifiability property by using the same arguments given for our construction with one authority.

Consider that w.l.o.g the $k$-th authority outputs a commitment $\mathsf{com}_k$ that does not commit to a tuple of 0's. If at least one voter $j$ is honest, the probability that this voter outputs a ballot $\mathsf{Blt}_j$ such that $\mathsf{com}_k$ is a commitment to a tuple $(z_1, \ldots, z_N)$ and $z_j = \mathsf{Blt}_j$ is negligible over the random coins of the $j$-th voter. Therefore, assuming that there is at least one honest voter, the authorities can compute proofs of tally correctness by using the witness for the "trapdoor statement" in the relation only with negligible probability. Similarly, assuming that there is at least one honest voter, the voters can compute proofs of ballot correctness by using the witness for the "trapdoor statement" only with negligible probability over the random coins of the honest voters. In conclusion, the generalized construction fulfills (a statistical variant of) the verifiability property thanks to the verifiability of our eVote in Section IV and to the fact that, in real mode, the sum of the messages encrypted in a ballot is equal to a number in $\{0, 1\}$.

### C. Privacy of the Construction

We use a selectively-secure model [65] for our definition of privacy. In the game between the challenger and the adversary, the adversary has to declare its challenge at the outset of the game before receiving the public keys of the authorities. The adversary is allowed to receive the secret keys of all except one authority.

We show that our generalized construction fulfills this definition of privacy. First, we define the sets $S^k$ and a method for computing the shares $v_{j,1}, \ldots, v_{j,m}$ for a vote $v_j$. This method must guarantee that any subset of $m-1$ authorities does not get any information about $v_j$. For simplicity, consider that $m = 2$. Then, the sets $S^1 = S^2 = S$ are equal by definition to $\{-p, \ldots, p\}$, where $p$ is a number of size super-polynomial in the security parameter. The message space of the PKE scheme must comprise numbers between $-Np$ and $Np$. To encrypt 0 (resp. 1), the voter chooses a random number $v_1$ in $S$ and sets $v_2$ to $-v_1$ (resp. $-v_1 + 1$). It is easy to see that, except when either $v_1$ or $v_2$ equal $-p$, any value of $v_2$ (resp. $v_1$) can correspond to $v_1 = -v_2$ (resp. $v_2 = -v_1$) if the voter cast a vote for 0 or to $v_1 = -v_2 + 1$ (resp. $v_2 = -v_1 + 1$) if the voter cast a vote for 1. The case in which either $v_1$ or $v_2$ equal $-p$ occurs with negligible probability, which is guaranteed by choosing $p$ to be super-polynomial in the security parameter. Consequently, each authority does not get any information on the vote $v_j$. This method can be generalized to the case $m > 2$. We skip the details.

Because the adversary receives the public keys after sending the challenge, in the security proof we can define a hybrid experiment where the commitments in the public key commit to ballots $(\mathsf{Blt}_1 \ldots, \mathsf{Blt}_N)$ computed on input the challenge messages. Like in the reduction of Section IV-C, we prove that the probability that the adversary submits an ill-formed ballot that is accepted by the ballot verification algorithm is negligible by using the computationally hiding property of the commitment scheme.

In the next hybrid experiments, the NIWI proofs of ballot correctness and of tally correctness can be computed by using the witness for the trapdoor statement, i.e., the randomness used to compute the commitments. Thanks to that, we are able to compute ballots where not all the ciphertexts encrypt the same message. This allows us to switch the message encrypted in one of the ciphertexts of the ballots and prove indistinguishability between the experiments by using the IND-CPA property of the PKE scheme.

To prove that our scheme fulfills a definition for privacy in a non-selective (i.e., full) security model, one can use complexity leveraging arguments. Such arguments can profit from the fact that, in our formulation, we required the number of voters $N$ and the size of the message space to be independent of the security parameter. This allows the challenger to just guess the challenge messages in advance with constant probability. This requirement can be weakened to the case of $N$ and size of message space logarithmic in the security parameter. We leave open how to achieve full security without complexity leveraging.

Note that we do not require any interaction between the authorities. The public keys of the authorities are completely *independent* from each other. Moreover, the authorities do not need any *coordination* (e.g., to run sequentially), i.e., the tally can be computed and publicly verified from the output of each authority individually. Thus, our techniques diverge from previous approaches to the problem.

## VI. Related Work

Our work is inspired by the work of Badrinarayanan *et al.* [38], which puts forward the concept of verifiable functional encryption. (We note that the committing IBE of [66] can be seen as a weaker variant of verifiable identity-based encryption.) Our work shares with BGJS the idea of "engineering" multiple witnesses, which are needed when using NIWI proofs, to enforce privacy in conjunction with verifiability.

Notwithstanding, the constructions are quite different, especially due to the different requirements of functional encryption and e-voting. For instance, in the security definition of functional encryption, the keys are handed to the adversary, so one needs a proof that each secret key and ciphertext is computed correctly. Instead, in our case, the adversary does not see the secret key. We can profit from this fact to just prove that the claimed tally equals the evaluation of the tally function over all ballots.

Such complications in functional encryption introduce a severe limitation: in the security reduction of BGJS, it is fundamental that the public key contain a commitment that in some hybrid experiment is set to the challenge ciphertext. Therefore, it is assumed that the adversary commits to the challenge before receiving the public key, i.e., security is proven in the *selective* model [65]. On the contrary, our constructions are secure in the *full* (i.e., non-selective) model.

In other respects, in e-voting we face new challenges. In BGJS, the challenger computes the NIWI on input a witness that comprises *all* the secret keys and proves the well-formedness of all the secret keys except one, but, in addition, proves that all the secret keys decrypt some challenge ciphertext correctly. This is sufficient to use the IND-CPA property of functional encryption to prove indistinguishability between two hybrid experiments where the message encrypted in one of the ciphertexts is switched from $m_0$ to $m_1$. The reason is that the secret keys are supposed to be for the same function $f$ such that $f(m_0) = f(m_1)$. (More concretely, in the IND-CPA property of functional encryption, the adversary is allowed to receive secret keys for a function $f$ that evaluates both challenge messages to the same value.) Therefore, the secret keys do not allow to distinguish between the two ciphertexts. In our setting, we can *only* input to the NIWI all the secret keys except one. Otherwise we could not use the IND-CPA property to prove indistinguishability between two hybrid experiments where the encrypted message is switched from $m_0$ to $m_1$.

Furthermore, in our privacy definition, we have to handle challenge tuples that contain ill-formed ballots, whereas in verifiable multi-input functional encryption the challenge contains only honestly computed ciphertexts. Therefore, the differences between the two settings make the respective techniques utterly incomparable.

It is tempting to think that the construction of BGJS of multi-input verifiable functional encryption (which extends multi-input functional encryption of Goldwasser *et al.* [67]) can be directly used to construct a verifiable eVote. Though it seems plausible, we did not verify that. However, this would eventually result in a verifiable eVote based on indistinguishability obfuscation [37], a very strong assumption, and would only be secure in the selective model.

Needless to say, our techniques, as well as the ones of BGJS, owe a lot to the celebrated FLS' OR trick [57]. They can be viewed as a generalization of it.

Kiayias *et al.* [68] (see also [69] for a distributed implementation) put forth a verifiable eVote without trust assumptions (except the existence of a PBB) that represents a breakthrough along this direction, but diverges from ours in several fundamental aspects:

- The definition of privacy in [68] includes receipt-freeness. To provide receipt-freeness, the construction in [68] requires untappable channels between voters and the authority.
- The definition of verifiability in [68] considers eligibility and end-to-end verifiability. The latter provides protection against adversarial voting devices.
- Our construction provide universal verifiability, i.e., external parties that do not participate in the elections can verify the result by checking the information published in the PBB, without requiring any information from protocol participants. In [68], for end-to-end verifiability, an external party needs to collect receipts from voters and then check the PBB. For universal verifiability, those receipts would not be needed.
- The universal verifiability achieved by our construction is perfect. The scheme in [68] provides end-to-end verifiability (including eligibility) information theoretically, but there is a small probability of a wrong tally being accepted that depends on the number of honest successful voters.
- In [68], the privacy property is proven under group-based assumptions at the cost of using complexity leveraging and assuming sub-exponential security, whereas ours only requires the standard version of Decision Linear

assumption with polynomial security.[7]

In a subsequent work [70], Kiayias *et al.* propose an e-voting scheme related to [68], which uses Groth-Sahai proofs and improves [68] in terms of efficiency. The scheme in [70] provides end-to-end verifiability without trust assumptions (except the existence of a PBB), but against computationally bounded adversaries.

Moran and Naor [71] construct an universally verifiable e-voting protocol with very strong provable-security properties. However, it assumes either the availability of a "random beacon" that has to be sampled honestly or the soundness of the Fiat-Shamir's heuristic. Therefore, verifiability does not hold unconditionally, i.e., without any assumption (both physical or computational).

We are not aware of other traditional e-voting schemes that achieve perfect verifiability without interaction and without trust assumptions. We refer to [72] and [54] for a survey.

We point out that our definition of verifiability is motivated by the guidelines of [72]. In its formalization, our definition is similar to the ones of [53], the verifiability for multi-input functional encryption of BGJS and the uniqueness of tally of Bernhard *et al.* [54]. Anyhow, the latter is formulated to hold only against computationally bounded adversaries and both BGJS16 and Bernhard *et al.* do not take into account our condition (2) for verifiability.[8] See also [73] for symbolic approaches to verifiability.

Perfect verifiability and perfect correctness seem incompatible with receipt-freeness [71], [74]–[78]. Notwithstanding, we think that it should be possible to define a statistical variant of verifiability achievable without any trust assumptions that could coexist with some form of receipt-freeness. Another possibility could be to resort to some voting server trusted for receipt-freeness but not for privacy, such as the server that re-randomizes the ballots in BeleniosRF of Chaidos, Cortier, Fuchsbauer and Galindo [78]. (We note that they also address the problem of authenticity that we neglect.) As it is out of the scope of this work, we deliberately omit receipt-freeness in our treatment.

## VII. Future Directions

Our work opens up new directions in e-voting and generally in cryptography. We discuss some of them.

- **Efficiency.** An important problem is to improve the efficiency of verification. It would be desirable that the cost for verifiers be sub-linear in the number of voters. The verifiability guarantees attained would then be computational but hopefully it could be possible to avoid trust assumptions. A possibility would be to employ variants of succinct arguments (see [79] for a survey).
- **Receipt-freeness.** Perfect verifiability and perfect correctness seem incompatible with receipt-freeness [71], [74]–[78], but we think that it should be possible to define a statistical variant of verifiability that could coexist with some form of receipt-freeness. Another possibility could be to resort to some voting server trusted for receipt-freeness but not for privacy that re-randomizes the ballots, as done in BeleniosRF of Chaidos, Cortier, Fuchsbauer and Galindo [78].
- **Other applications of our techniques.** We think that our techniques could be of wide applicability to other settings. For instance, Camenisch and Shoup [80] put forth the concept of verifiable encryption (that in some sense could be also viewed as a special case of verifiable functional encryption [38]) and present numerous applications of it, such as key escrow, optimistic fair exchange, publicly verifiable secret and signature sharing, universally composable commitments, group signatures, and confirmer signatures. We believe that our techniques can be employed profitably to improve their results with the aim of removing the need of trust assumptions.

**Acknowledgements.** Vincenzo Iovino was supported by the Luxembourg National Research Fund (FNR grant no. 7884937). Alfredo Rial is supported by the Luxembourg National Research Fund (FNR) CORE project "Stateful Zero-Knowledge" (Project code: C17/11650748). This work was also supported by the INTER-Sequoia project from the Luxembourg National Research Fund, which is joint with the ANR project SEQUOIA ANR-14-CE28-0030-01.

[7] At some point in the security reduction for our eVote, we make use of the fact that the number of voters $N$ is a constant independent of the security parameter that could be viewed as a complexity leveraging trick or as problematic in the case that $N$ be large. But we stress that this is done only for simplicity of exposition and we sketch how the reduction and our results can be generalized even to the case of $N(\cdot)$ function of the security parameter.

[8] Needless to say, for many applications of multi-input functional encryption, the lack of condition (2) could not pose a threat.

## References

[1] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM*, vol. 24, no. 2, pp. 84–90, 1981.

[2] R. Cramer, R. Gennaro, and B. Schoenmakers, "A secure and optimally efficient multi-authority election scheme," in *Advances in Cryptology – EUROCRYPT'97*, ser. Lecture Notes in Computer Science, W. Fumy, Ed., vol. 1233. Springer, May 1997, pp. 103–118.

[3] I. Damgård and M. Jurik, "A generalisation, a simplification and some applications of Paillier's probabilistic public-key system," in *PKC 2001: 4th International Workshop on Theory and Practice in Public Key Cryptography*, ser. Lecture Notes in Computer Science, K. Kim, Ed., vol. 1992. Springer, Feb. 2001, pp. 119–136.

[4] P. Y. A. Ryan and S. A. Schneider, "Prêt à voter with re-encryption mixes," University of Newcastle, Tech. Rep. CS-TR-956, 2006.

[5] B. Adida, "Helios: Web-based open-audit voting." in *USENIX Security Symposium*, vol. 17, 2008, pp. 335–348.

[6] D. Chaum, R. Carback, J. Clark, A. Essex, S. Popoveniuc, R. L. Rivest, P. Y. A. Ryan, E. Shen, A. T. Sherman, and P. L. Vora, "Scantegrity II: end-to-end verifiability by voters of optical scan elections through confirmation codes," *IEEE Trans. Information Forensics and Security*, vol. 4, no. 4, pp. 611–627, 2009. [Online]. Available: http://dx.doi.org/10.1109/TIFS.2009.2034919

[7] P. Y. A. Ryan and V. Teague, "Pretty good democracy," in *IN: WORKSHOP ON SECURITY PROTOCOLS*, 2009.

[8] A. Juels, D. Catalano, and M. Jakobsson, "Coercion-resistant electronic elections," in *Towards Trustworthy Elections*. Springer, 2010, pp. 37–63.

[9] M. Blum, P. Feldman, and S. Micali, "Non-interactive zero-knowledge and its applications (extended abstract)," in *20th Annual ACM Symposium on Theory of Computing*. ACM Press, May 1988, pp. 103–112.

[10] A. De Santis, S. Micali, and G. Persiano, "Non-interactive zero-knowledge proof systems," in *Advances in Cryptology – CRYPTO'87*, ser. Lecture Notes in Computer Science, C. Pomerance, Ed., vol. 293. Springer, Aug. 1988, pp. 52–72.

[11] C. Rackoff and D. R. Simon, "Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack," in *Advances in Cryptology – CRYPTO'91*, ser. Lecture Notes in Computer Science, J. Feigenbaum, Ed., vol. 576. Springer, Aug. 1992, pp. 433–444.

[12] O. Goldreich, *Foundations of Cryptography: Basic Techniques*. Cambridge, UK: Cambridge University Press, 2001, vol. 1.

[13] A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai, "Robust non-interactive zero knowledge," in *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, 2001, pp. 566–598.

[14] M. Naor and M. Yung, "Public-key cryptosystems provably secure against chosen ciphertext attacks," in *22nd Annual ACM Symposium on Theory of Computing*. ACM Press, May 1990, pp. 427–437.

[15] R. Cramer and V. Shoup, "Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack," *SIAM Journal on Computing*, vol. 33, no. 1, pp. 167–226, 2003.

[16] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *ACM CCS 93: 1st Conference on Computer and Communications Security*, V. Ashby, Ed. ACM Press, Nov. 1993, pp. 62–73.

[17] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Advances in Cryptology – CRYPTO'86*, ser. Lecture Notes in Computer Science, A. M. Odlyzko, Ed., vol. 263. Springer, Aug. 1987, pp. 186–194.

[18] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche, "The KECCAK reference," 2011, http://keccak.noekeon.org/.

[19] R. Canetti, O. Goldreich, and S. Halevi, "The random oracle methodology, revisited (preliminary version)," in *30th Annual ACM Symposium on Theory of Computing*. ACM Press, May 1998, pp. 209–218.

[20] S. Goldwasser and Y. T. Kalai, "On the (in)security of the Fiat-Shamir paradigm," in *44th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, Oct. 2003, pp. 102–115.

[21] Y. T. Kalai, "Attacks on the fiat-shamir paradigm and program obfuscation," Ph.D. dissertation, Massachusetts Institute of Technology, 2006.

[22] N. Bitansky, D. Dachman-Soled, S. Garg, A. Jain, Y. T. Kalai, A. López-Alt, and D. Wichs, "Why "fiat-shamir for proofs" lacks a proof," in *Theory of Cryptography: 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013*. Springer, 2013, pp. 182–201.

[23] I. Damgård, N. Fazio, and A. Nicolosi, "Non-interactive zero-knowledge from homomorphic encryption," in *TCC 2006: 3rd Theory of Cryptography Conference*, ser. Lecture Notes in Computer Science, S. Halevi and T. Rabin, Eds., vol. 3876. Springer, Mar. 2006, pp. 41–59.

[24] Y. Lindell, "An efficient transform from sigma protocols to NIZK with a CRS and non-programmable random oracle," in *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part I*, 2015, pp. 93–109.

[25] P. Chaidos and J. Groth, "Making sigma-protocols non-interactive without random oracles," in *Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings*, 2015, pp. 650–670.

[26] M. Ciampi, G. Persiano, L. Siniscalchi, and I. Visconti, "A transform for NIZK almost as efficient and general as the fiat-shamir transform without programmable random oracles," in *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part II*, 2016, pp. 83–111.

[27] H. Lipmaa, "Secure electronic voting protocols," in *Handbook of Information Security, Volume 2, Information Warfare, Social, Legal, and International Issues and Security Foundations*, H. Bidgoli, Ed. John Wiley & Sons, Inc., 2005, pp. 647–657, electronic edition available at http://kodu.ut.ee/ lipmaa/papers/voting4hb.pdf.

[28] J. Groth, R. Ostrovsky, and A. Sahai, "Non-interactive zaps and new techniques for NIZK," in *Advances in Cryptology – CRYPTO 2006*, ser. Lecture Notes in Computer Science, C. Dwork, Ed., vol. 4117. Springer, Aug. 2006, pp. 97–111.

[29] C. Dwork and M. Naor, "Zaps and their applications," in *41st Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, Nov. 2000, pp. 283–293.

[30] B. Barak, S. J. Ong, and S. P. Vadhan, "Derandomization in cryptography," in *Advances in Cryptology – CRYPTO 2003*, ser. Lecture Notes in Computer Science, D. Boneh, Ed., vol. 2729. Springer, Aug. 2003, pp. 299–315.

[31] N. Bitansky and O. Paneth, "Zaps and non-interactive witness indistinguishability from indistinguishability obfuscation," in *Theory of Cryptography Conference*. Springer, 2015, pp. 401–427.

[32] J. Groth and A. Sahai, "Efficient non-interactive proof systems for bilinear groups," in *Advances in Cryptology – EUROCRYPT 2008*, ser. Lecture Notes in Computer Science, N. P. Smart, Ed., vol. 4965. Springer, Apr. 2008, pp. 415–432.

[33] M. Bellare, G. Fuchsbauer, and A. Scafuro, "Nizks with an untrusted CRS: security in the face of parameter subversion," in *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II*, 2016, pp. 777–804.

[34] B. Abdolmaleki, K. Baghery, H. Lipmaa, and M. Zajkac, "A subversion-resistant snark," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2017, pp. 3–33.

[35] G. Fuchsbauer, "Subversion-zero-knowledge snarks," in *IACR International Workshop on Public Key Cryptography*. Springer, 2018, pp. 315–347.

[36] D. Boneh, A. Sahai, and B. Waters, "Functional encryption: Definitions and challenges," in *TCC 2011: 8th Theory of Cryptography Conference*, ser. Lecture Notes in Computer Science, Y. Ishai, Ed., vol. 6597. Springer, Mar. 2011, pp. 253–273.

[37] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters, "Candidate indistinguishability obfuscation and functional encryption for all circuits," in *54th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, Oct. 2013, pp. 40–49.

[38] S. Badrinarayanan, V. Goyal, A. Jain, and A. Sahai, "Verifiable functional encryption," in *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II*, 2016, pp. 557–587.

[39] D. Boneh and M. K. Franklin, "Identity based encryption from the Weil pairing," *SIAM Journal on Computing*, vol. 32, no. 3, pp. 586–615, 2003.

[40] A. Joux, "A one round protocol for tripartite Diffie-Hellman," *Journal of Cryptology*, vol. 17, no. 4, pp. 263–276, Sep. 2004.

[41] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Advances in Cryptology – CRYPTO 2004*, ser. Lecture Notes in Computer Science, M. Franklin, Ed., vol. 3152. Springer, Aug. 2004, pp. 41–55.

[42] M. Naor, "On cryptographic assumptions and challenges (invited talk)," in *Advances in Cryptology – CRYPTO 2003*, ser. Lecture Notes in Computer Science, D. Boneh, Ed., vol. 2729. Springer, Aug. 2003, pp. 96–109.

[43] E. Ghadafi, N. P. Smart, and B. Warinschi, "Groth-Sahai proofs revisited," in *PKC 2010: 13th International Conference on Theory and Practice of Public Key Cryptography*, ser. Lecture Notes in Computer Science, P. Q. Nguyen and D. Pointcheval, Eds., vol. 6056. Springer, May 2010, pp. 177–192.

[44] J. Groth and R. Ostrovsky, "Cryptography in the multi-string model," *Journal of Cryptology*, vol. 27, no. 3, pp. 506–543, Jul. 2014.

[45] R. L. Rivest, "The threeballot voting system," 2006.

[46] B. Randell and P. Y. A. Ryan, "Voting technologies and trust," in *IEEE Security and Privacy*, 2006, pp. 50–56.

[47] P. Y. A. Ryan, P. B. Rønne, and V. Iovino, "Selene: Voting with transparent verifiability and coercion-mitigation," in *Financial Cryptography and Data Security - FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers*, 2016, pp. 176–192.

[48] A. Kiayias and M. Yung, "Self-tallying elections and perfect ballot secrecy," in *PKC 2002: 5th International Workshop on Theory and Practice in Public Key Cryptography*, ser. Lecture Notes in Computer Science, D. Naccache and P. Paillier, Eds., vol. 2274. Springer, Feb. 2002, pp. 141–158.

[49] I. Damgård and M. Jurik, "A length-flexible threshold cryptosystem with applications," in *ACISP 03: 8th Australasian Conference on Information Security and Privacy*, ser. Lecture Notes in Computer Science, R. Safavi-Naini and J. Seberry, Eds., vol. 2727. Springer, Jul. 2003, pp. 350–364.

[50] J. Groth, "Efficient maximal privacy in boardroom voting and anonymous broadcast," in *International Conference on Financial Cryptography*. Springer, 2004, pp. 90–104.

[51] F. Hao, P. Y. A. Ryan, and P. Zielinski, "Anonymous voting by two-round public discussion," *IET Information Security*, vol. 4, no. 2, pp. 62–67, 2010.

[52] D. Khader, B. Smyth, P. Y. A. Ryan, and F. Hao, "A fair and robust voting system by broadcast," in *5th International Conference on Electronic Voting 2012, (EVOTE 2012), Co-organized by the Council of Europe, Gesellschaft für Informatik and E-Voting.CC, July 11-14, 2012, Castle Hofen, Bregenz, Austria*, 2012, pp. 285–299.

[53] R. Giustolisi, V. Iovino, and P. Rønne, "On the possibility of non-interactive voting in the public-key setting," in *Financial Cryptography and Data Security - FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers*, 2016.

[54] D. Bernhard, V. Cortier, D. Galindo, O. Pereira, and B. Warinschi, "Sok: A comprehensive analysis of game-based ballot privacy definitions," in *2015 IEEE Symposium on Security and Privacy*. IEEE, 2015, pp. 499–516.

[55] V. Cortier and B. Smyth, "Attacking and fixing helios: An analysis of ballot secrecy," Cryptology ePrint Archive, Report 2010/625, 2010, http://eprint.iacr.org/2010/625.

[56] J. Benaloh, "Verifiable secret-ballot elections," Ph.D. dissertation, Yale University, 1987.

[57] U. Feige, D. Lapidot, and A. Shamir, "Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract)," in *31st Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, Oct. 1990, pp. 308–317.

[58] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[59] A. C.-C. Yao, "How to generate and exchange secrets (extended abstract)," in *27th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, Oct. 1986, pp. 162–167.

[60] Y. Lindell and B. Pinkas, "A proof of security of Yao's protocol for two-party computation," *Journal of Cryptology*, vol. 22, no. 2, pp. 161–188, Apr. 2009.

[61] S. Goldwasser, Y. T. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich, "Reusable garbled circuits and succinct functional encryption," in *45th Annual ACM Symposium on Theory of Computing*, D. Boneh, T. Roughgarden, and J. Feigenbaum, Eds. ACM Press, Jun. 2013, pp. 555–564.

[62] A. Sahai and H. Seyalioglu, "Worry-free encryption: functional encryption with public keys," in *ACM CCS 10: 17th Conference on Computer and Communications Security*, E. Al-Shaer, A. D. Keromytis, and V. Shmatikov, Eds. ACM Press, Oct. 2010, pp. 463–472.

[63] S. Gorbunov, V. Vaikuntanathan, and H. Wee, "Functional encryption with bounded collusions via multi-party computation," in *Advances in Cryptology – CRYPTO 2012*, ser. Lecture Notes in Computer Science, R. Safavi-Naini and R. Canetti, Eds., vol. 7417. Springer, Aug. 2012, pp. 162–179.

[64] S. Garg, C. Gentry, S. Halevi, and M. Zhandry, "Functional encryption without obfuscation," in *Theory of Cryptography: 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part II*, E. Kushilevitz and T. Malkin, Eds. Springer, 2016, pp. 480–511.

[65] R. Canetti, S. Halevi, and J. Katz, "Chosen-ciphertext security from identity-based encryption," in *Advances in Cryptology – EUROCRYPT 2004*, ser. Lecture Notes in Computer Science, C. Cachin and J. Camenisch, Eds., vol. 3027. Springer, May 2004, pp. 207–222.

[66] M. Green and S. Hohenberger, "Blind identity-based encryption and simulatable oblivious transfer," in *Advances in Cryptology – ASIACRYPT 2007*, ser. Lecture Notes in Computer Science, K. Kurosawa, Ed., vol. 4833. Springer, Dec. 2007, pp. 265–282.

[67] S. Goldwasser, S. D. Gordon, V. Goyal, A. Jain, J. Katz, F.-H. Liu, A. Sahai, E. Shi, and H.-S. Zhou, "Multi-input functional encryption," in *Advances in Cryptology – EUROCRYPT 2014*, ser. Lecture Notes in Computer Science, P. Q. Nguyen and E. Oswald, Eds., vol. 8441. Springer, May 2014, pp. 578–602.

[68] A. Kiayias, T. Zacharias, and B. Zhang, "End-to-end verifiable elections in the standard model," in *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, 2015, pp. 468–498.

[69] N. Chondros, B. Zhang, T. Zacharias, P. Diamantopoulos, S. Maneas, C. Patsonakis, A. Delis, A. Kiayias, and M. Roussopoulos, "A distributed, end-to-end verifiable, internet voting system," *CoRR*, vol. abs/1507.06812, 2015. [Online]. Available: http://arxiv.org/abs/1507.06812

[70] A. Kiayias, T. Zacharias, and B. Zhang, "DEMOS-2: scalable E2E verifiable elections without random oracles," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, 2015, pp. 352–363. [Online]. Available: https://doi.org/10.1145/2810103.2813727

[71] T. Moran and M. Naor, "Receipt-free universally-verifiable voting with everlasting privacy," in *Advances in Cryptology – CRYPTO 2006*, ser. Lecture Notes in Computer Science, C. Dwork, Ed., vol. 4117. Springer, Aug. 2006, pp. 373–392.

[72] V. Cortier, D. Galindo, R. Küsters, J. Mueller, and T. Truderung, "Sok: Verifiability notions for e-voting protocols," in *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, 2016, pp. 779–798.

[73] S. Kremer, M. Ryan, and B. Smyth, "Election verifiability in electronic voting protocols," in *European Symposium on Research in Computer Security*. Springer, 2010, pp. 389–404.

[74] J. C. Benaloh and D. Tuinstra, "Receipt-free secret-ballot elections (extended abstract)," in *26th Annual ACM Symposium on Theory of Computing*. ACM Press, May 1994, pp. 544–553.

[75] K. Sako and J. Kilian, "Receipt-free mix-type voting scheme - a practical solution to the implementation of a voting booth," in *Advances in Cryptology – EUROCRYPT'95*, ser. Lecture Notes in Computer Science, L. C. Guillou and J.-J. Quisquater, Eds., vol. 921. Springer, May 1995, pp. 393–403.

[76] M. Michels and P. Horster, "Some remarks on a receipt-free and universally verifiable mix-type voting scheme," in *Advances in Cryptology – ASIACRYPT'96*, ser. Lecture Notes in Computer Science, K. Kim and T. Matsumoto, Eds., vol. 1163. Springer, Nov. 1996, pp. 125–132.

[77] S. Delaune, S. Kremer, and M. Ryan, "Verifying privacy-type properties of electronic voting protocols," *Journal of Computer Security*, vol. 17, no. 4, pp. 435–487, 2009. [Online]. Available: http://dx.doi.org/10.3233/JCS-2009-0340

[78] P. Chaidos, V. Cortier, G. Fuchsbauer, and D. Galindo, "Beleniosrf: A non-interactive receipt-free electronic voting scheme," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, 2016, pp. 1614–1625, full version available at http://eprint.iacr.org/2015/629.

[79] N. Bitansky, "Getting inside the adversary's head: New directions in non-black-box knowledge extraction," Ph.D. dissertation, Tel Aviv University, 2014.

[80] J. Camenisch and V. Shoup, "Practical verifiable encryption and decryption of discrete logarithms," in *Advances in Cryptology – CRYPTO 2003*, ser. Lecture Notes in Computer Science, D. Boneh, Ed., vol. 2729. Springer, Aug. 2003, pp. 126–144.

[81] A. De Caro, V. Iovino, A. Jain, A. O'Neill, O. Paneth, and G. Persiano, "On the achievability of simulation-based security for functional encryption," in *Advances in Cryptology – CRYPTO 2013, Part II*, ser. Lecture Notes in Computer Science, R. Canetti and J. A. Garay, Eds., vol. 8043. Springer, Aug. 2013, pp. 519–535.

[82] J. Groth, "Simulation-sound nizk proofs for a practical language and constant size group signatures," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2006, pp. 444–459.

[83] J. Groth, R. Ostrovsky, and A. Sahai, "New techniques for noninteractive zero-knowledge," *Journal of the ACM (JACM)*, vol. 59, no. 3, p. 11, 2012.

[84] C. Ràfols, "Stretching groth-sahai: NIZK proofs of partial satisfiability," in *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, 2015, pp. 247–276.

## APPENDIX A
## BOOSTING OUR PRIVACY DEFINITION TO SIMULATION-BASED SECURITY

Our definition, though formulated as an indistinguishability-based one, can be bootstrapped, for large classes of tallying functions, to a simulation-based one. Consider a real and an ideal world defined as follows. In the real world, the adversary outputs a tuple of messages that specify the set $S$ of possibly

invalid ballots as in the definition in Fig. 3. The adversary gets a set of ballots corresponding to such messages, after replacing the invalid ballots with $\perp$, along with a proof $\gamma$ exactly as in the definition in Fig. 3. In the ideal world, in addition to the adversary, there is a simulator. The simulator gets as input the set of possibly invalid ballots and the evaluation of the tally function on the messages provided by the adversary, and has to output an indistinguishable set of ballots along with the proof.

The adversary has to guess whether the set of ballots along with the proof were simulated or computed correctly. We require that, for any PPT adversary, there exists a simulator such that the advantage of the adversary in distinguishing the real world from the ideal world is negligible.

Our IND-Secure definition implies such a simulation-based definition when the tally function is invertible (i.e. for almost all interesting e-voting applications) for the following reason. For concreteness, let us focus on the case of the sum function. The simulator does not know the messages output by the adversary but gets as input the set of possibly invalid ballots and, in addition, the result of the sum function. Therefore the simulator, for the indices corresponding to valid ballots, can define messages that sum up to the desired result (that it knows) and, by using the public key of the authority, can encrypt them and add a proof to the ballot. (It can do that because it knows the corresponding witness). Note that, for the indices corresponding to possibly invalid ballots, the simulator can just check by using the verification ballot algorithm whether the ballot is invalid and replace it by $\perp$. Thanks to the IND-Security, the view created by the simulator will be indistinguishable to the view in the real world.

It is easy to see that this can be generalized to any invertible function. This is similar to the way De Caro *et al.* [81] amplify IND-Security to SIM-Security for functional encryption.

## APPENDIX B
## BUILDING BLOCKS

Our construction uses IND-CPA public-key encryption with perfect correctness and unique secret key, perfectly binding commitment schemes, and (one-message) non-interactive witness-indistinguishable proof systems with perfect soundness for NP [28] (see also [29], [29]–[31], [57]). In this section, we recall the definitions of those primitives.

### A. Public-Key Encryption

*Definition B.1:* [IND-CPA secure PKE with perfect correctness and unique secret key] An IND-CPA (or semantically) secure Public-Key Encryption (PKE) scheme consists of three PPT algorithms $(\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt})$ defined as follows.

- $\mathsf{Setup}(1^\lambda)$: On input $1^\lambda$, it outputs public key $\mathsf{Pk}$ and decryption key $\mathsf{Sk}$.
- $\mathsf{Encrypt}(m, \mathsf{Pk})$: On input message $m$ and public key $\mathsf{Pk}$, it outputs ciphertext $\mathsf{Ct}$.
- $\mathsf{Decrypt}(\mathsf{Ct}, \mathsf{Sk})$: On input ciphertext $\mathsf{Ct}$ and decryption key $\mathsf{Sk}$, it outputs $m$.

The PKE scheme is said to be IND-CPA (or semantically) secure if for any PPT adversary $\mathcal{A}$, there exists a negligible function $\nu(\cdot)$ such that the following is satisfied for any two messages $m_0, m_1$ and for $b \in \{0, 1\}$:

$$|\Pr\left[\mathcal{A}(1^\lambda, \mathsf{Encrypt}(m_0, \mathsf{Pk})) = b\right] - \Pr\left[\mathcal{A}(1^\lambda, \mathsf{Encrypt}(m_1, \mathsf{Pk})) = b\right]| \leq \nu(\lambda).$$

*Perfect correctness* requires that, for all pairs $(\mathsf{Pk}, \mathsf{Sk}) \in \mathsf{Setup}$, for all messages $m$ in the message space and all ciphertexts $\mathsf{Ct}$ output by $\mathsf{Encrypt}(\mathsf{Pk}, m)$, $\mathsf{Decrypt}(\mathsf{Ct}, \mathsf{Sk}) = m$ must hold. *Unique secret key* requires that, for all $\mathsf{Pk}$, there exists at most *one* $\mathsf{Sk}$ such that $(\mathsf{Pk}, \mathsf{Sk}) \in \mathsf{Setup}(1^\lambda)$.

The Decision Linear Encryption scheme [41] fulfills those properties. It is secure under the Decision Linear Assumption [41]. We recall them next.

First, we define bilinear groups. We assume the existence of a PPT algorithm $\mathcal{G}(1^\lambda)$, the *bilinear group generator*, that outputs a *pairing group setup* $(p, \mathbb{G}, \mathbb{G}_t, \mathbf{e}, g)$, where $\mathbb{G}$ and $\mathbb{G}_t$ are multiplicative groups of prime order $p$ and $\mathbf{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_t$ is a *bilinear map* satisfying the following three properties: (1) bilinearity, i.e., $\mathbf{e}(g^x, g^y) = e(g, g)^{xy}$; (2) non-degeneracy, i.e., for all generators $g \in \mathbb{G}$, $\mathbf{e}(g, g)$ generates $\mathbb{G}_t$; (3) efficiency, i.e., $\mathbf{e}$ can be computed in polynomial time.

*Assumption 1 (Decision Linear Assumption for $\mathcal{G}$. [41]):* Let the tuple $(p, \mathbb{G}, \mathbb{G}_t, \mathbf{e}, g)$ be a pairing group setup output by $\mathcal{G}$ as defined above, and let $g_1$, $g_2$ and $g_3$ be generators of $\mathbb{G}$. Given $(g_1, g_2, g_3, g_1^a, g_2^b, g_3^c)$, where $a$ and $b$ are picked randomly from $\mathbb{Z}_p$, the Decision Linear (DLIN) assumption is to decide whether $c = a + b \bmod p$. Precisely, the advantage of an adversary $\mathcal{A}$ in solving the Decision Linear assumption is given by:

$$\begin{aligned}&\big|\Pr\left[\mathcal{A}(\mathbb{G}, p, g_1, g_2, g_3, g_1^a, g_2^b, g_3^{a+b}) = 1 \mid (p, \mathbb{G}, \mathbb{G}_t, \mathbf{e}, g) \leftarrow \mathcal{G}(1^\lambda); (g_1, g_2, g_3) \leftarrow \mathbb{G}; (a, b) \leftarrow \mathbb{Z}_p\right] - \\ &\Pr\left[\mathcal{A}(\mathbb{G}, p, g_1, g_2, g_3, g_1^a, g_2^b, g_3^c) = 1 \mid (p, \mathbb{G}, \mathbb{G}_t, \mathbf{e}, g) \leftarrow \mathcal{G}(1^\lambda); (g_1, g_2, g_3) \leftarrow \mathbb{G}; (a, b, c) \leftarrow \mathbb{Z}_p\right]\big|\end{aligned}$$

The Decision Linear assumption states that the advantage of $\mathcal{A}$ is negligible in $\lambda$. Boneh *et al.* [41] provide a bilinear group generator $\mathcal{G}$ for which such assumption is conjectured to hold.

Consider the following PKE scheme described by a setup algorithm Setup, an encryption algorithm Encrypt and a decryption algorithm Decrypt.

- $\mathsf{Setup}(1^\lambda)$: pick $(p, \mathbb{G}, \mathbb{G}_t, \mathbf{e}, g) \leftarrow \mathcal{G}(1^\lambda)$, pick randomly $(x, y) \leftarrow \mathbb{Z}_p$. Compute $f = g^{1/x}$ and $h = g^{1/y}$. Output the public key $\mathsf{Pk} = (\mathbb{G}, p, g, f, h)$ and the secret key $\mathsf{Sk} = (\mathsf{Pk}, x, y)$.
- $\mathsf{Encrypt}(\mathsf{Pk}, m)$: on input a public key $\mathsf{Pk}$ and a message $m \in \mathbb{G}$, pick random $(a, b) \leftarrow \mathbb{Z}_p$. Output a ciphertext $\mathsf{Ct} = (f^a, h^b, m \cdot g^{a+b})$.
- $\mathsf{Decrypt}(\mathsf{Sk}, \mathsf{Ct})$: on input a secret key $\mathsf{Sk}$ and a ciphertext $\mathsf{Ct} = (c_1, c_2, c_3)$, output $m = c_3/(c_1^x c_2^y)$.

This scheme fulfills the IND-CPA property under the Decision Linear assumption (see [41] for details) and it is easy to verify that it fulfills the unique secret key property.

### B. Commitment Schemes

*Definition B.2:* [(Perfectly binding) Commitment Schemes] A commitment scheme $\mathsf{Com}$ is a PPT algorithm that takes as input a string $x$ and randomness $r \in \{0,1\}^k$ and outputs $\mathsf{com} \leftarrow \mathsf{Com}(x;r)$. A perfectly binding commitment scheme must satisfy the following properties:

- Perfectly Binding: This property states that two different strings cannot have the same commitment. More formally, $\forall x_1 \neq x_2$ and $r_1, r_2, \mathsf{Com}(x_1;r_1) \neq \mathsf{Com}(x_2;r_2)$.
- Computational Hiding: For all strings $x_0$ and $x_1$ (of the same length), there exists a negligible function $\nu(\cdot)$ such that, for all PPT adversaries $\mathcal{A}$, we have that the following holds: $|\Pr_{r\in\{0,1\}^k}[\,\mathcal{A}(\mathsf{Com}(x_0;r))=1\,] - \Pr_{r\in\{0,1\}^k}[\,\mathcal{A}(\mathsf{Com}(x_1;r))=1)\,]| \leq \nu(k)$.

### C. NIWI Proofs

We define (one-message) non-interactive witness indistinguishability (NIWI) proof systems [28]. Groth *et al.* [28] construct such NIWIs for all languages in NP, and in particular for CircuitSat.

*Definition B.3:* [Non-interactive Proof System] A non-interactive proof system for a language $L$ with a PPT relation $R$ is a tuple of algorithms $(\mathsf{Prove}, \mathsf{Verify})$. Prove receives as input a statement $x$ and a witness $w$ and outputs a proof $\pi$. Verify receives as input a statement $x$ and a proof $\pi$ and outputs a symbol in $\{\mathsf{OK}, \perp\}$. The following properties must hold:

- Perfect Completeness: For every $(x,w) \in R$, it holds that $\Pr[\,\mathsf{Verify}(x, \mathsf{Prove}(x,w)) = \mathsf{OK}\,] = 1$, where the probability is taken over the coins of Prove and Verify.
- Perfect Soundness: For every family $\{x_k\}_{k>0}$ of statements $x_k \notin L, x \in \{0,1\}^k$, for every non-uniform (possibly, computationally unbounded) adversary $\mathcal{A} = \{\mathcal{A}_k\}_{k>0}$, for all $k > 0$, it holds that:

$$\Pr\left[\begin{array}{l}\mathsf{Verify}(x_k,\pi) = \mathsf{OK}: \\ \pi \leftarrow \mathcal{A}_k(x_k)\end{array}\right] = 0.$$

*Definition B.4:* [(one-message) NIWI proof system] A non-interactive proof system $\mathsf{NIWI} = (\mathsf{Prove}, \mathsf{Verify})$ for a language $L$ with a PPT relation $R$ is witness-indistinguishable (WI) if for any triplet $(x, w_0, w_1)$ such that $(x, w_0) \in R$ and $(x, w_1) \in R$, the distributions $\{\mathsf{Prove}(x, w_0)\}$ and $\{\mathsf{Prove}(x, w_1)\}$ are computationally indistinguishable.

## APPENDIX C
## CORRECTNESS AND VERIFIABILITY OF OUR EVOTE

### A. Correctness of Our eVote

Condition (1) of (perfect) correctness of EVOTE follows from the perfect correctness of the PKE scheme and the perfect completeness of $\mathsf{NIWI}^{\mathsf{enc}}$ and $\mathsf{NIWI}^{\mathsf{dec}}$. Condition (2) follows analogously. We note the following. For all honestly computed $\mathsf{Pk}$, $\mathsf{Pk} = (\mathsf{Pk}_1, \mathsf{Pk}_2, \mathsf{Pk}_3, Z)$ holds for some $\mathsf{Pk}_1, \mathsf{Pk}_2, \mathsf{Pk}_3$ and $Z$. $Z$ is a commitment to 1. Therefore, relation $\mathsf{R}^{\mathsf{enc}}$ and the perfectly binding property of the commitment scheme imply that, if there exists a proof $\pi$ and a statement $x = (j, \mathsf{Ct}_1, \ldots, \mathsf{Ct}_3, \mathsf{Pk}_1, \ldots, \mathsf{Pk}_3, Z)$ such that VerifyBallot accepts $(x, \pi)$, then it must be the case that $\mathsf{Ct}_1, \ldots, \mathsf{Ct}_3$ encrypt the same string in $\mathcal{M} \cup \{\perp\}$. For all $j \in [N]$, if $\mathsf{Blt}_j$ is accepted by VerifyBallot, $\mathsf{Blt}'_j = \mathsf{Blt}_j$, else $\mathsf{Blt}'_j = \perp$. Therefore, for all $\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N$, if $(y, \gamma) = \mathsf{EvalTally}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N)$, then $y = F(m_1, \ldots, m_N)$, where, for all $j \in [N]$, if $\mathsf{Blt}_j$ is accepted by VerifyBallot, $m_j$ is the string encrypted in the first two ciphertexts of $\mathsf{Blt}_j$, else $m_j$ is $\perp$. Then, it is easy to see that $\mathsf{VerifyTally}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y, \gamma) = \mathsf{OK}$.

### B. Verifiability of Our eVote

*Theorem C.1:* For all $N > 0$, all sets $\mathcal{M}, \Sigma \subset \{0,1\}^\star$, and all tally functions $F : (\mathcal{M} \cup \{\perp\})^N \rightarrow \Sigma \cup \{\perp\}$, if $\mathcal{E}$ is a perfectly correct PKE scheme with unique secret key (cf. Def. B.1), Com is a PPT algorithm, and $\mathsf{NIWI}^{\mathsf{enc}}$ and $\mathsf{NIWI}^{\mathsf{dec}}$ are (one-message) NIWIs (cf. Def. B.4), for the relations $\mathsf{R}^{\mathsf{enc}}$ and $\mathsf{R}^{\mathsf{dec}}$ respectively, then $\mathsf{EVOTE}^{N,\mathcal{M},\Sigma,F,\mathcal{E},\mathsf{Com},\mathsf{NIWI}^{\mathsf{enc}},\mathsf{NIWI}^{\mathsf{dec}}}$ satisfies the verifiability property (cf. Def. 3).

**Proof:** We first prove that condition (1) of verifiability is satisfied. We have to prove that, for all $\mathsf{Pk} \in \{0,1\}^\star$, and all $\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N \in \{0,1\}^\star \cup \{\perp\}$ such that, for all $j \in [N]$, either $\mathsf{Blt}_j = \perp$ or $\mathsf{VerifyBallot}(\mathsf{Pk}, j, \mathsf{Blt}_j) = \mathsf{OK}$, there exist $m_1, \ldots, m_N \in \mathcal{M} \cup \{\perp\}$ such that, for all $y, \gamma \in \{0,1\}^\star$, if $\mathsf{VerifyTally}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y, \gamma) = 1$ then $y = F(m_1, \ldots, m_N)$. Henceforth, w.l.o.g, we let $\mathsf{Pk}$ and $\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N$ be arbitrary strings such that, for all $j \in [N]$, either $\mathsf{Blt}_j = \perp$ or $\mathsf{VerifyBallot}(\mathsf{Pk}, j, \mathsf{Blt}_j) = \mathsf{OK}$.

First, we prove the following claim.

*Claim C.2:* Given $\mathsf{Pk}$ and $(\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N)$, for every two pairs $(y_0, \gamma_0)$ and $(y_1, \gamma_1)$, if $\mathsf{VerifyTally}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y_0, \gamma_0) = \mathsf{VerifyTally}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y_1, \gamma_1) = \mathsf{OK}$ then $y_0 = y_1$.

For every $(y_0, \gamma_0)$ and $(y_1, \gamma_1)$, we have two cases.

1) Either $y_0 = \perp$ and $y_1 \neq \perp$ or $y_1 = \perp$ and $y_0 \neq \perp$. Suppose w.l.o.g. that $y_0 = \perp$ and $y_1 \neq \perp$. The other case (i.e., $y_1 = \perp$ and $y_0 \neq \perp$) is symmetrical.
   By construction, for all $(y, \gamma)$, it holds that (A) if $\mathsf{Blt}_1 = \cdots = \mathsf{Blt}_N = \perp$, then $\mathsf{VerifyTally}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y, \gamma) = \mathsf{OK}$ if and only if $y = \perp$ and (B) if, for some $j \in [N], \mathsf{Blt}_j \neq \perp$, then $\mathsf{VerifyTally}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, \perp, \gamma) = \perp$. We now have two cases.
   a) $\mathsf{Blt}_1 = \cdots = \mathsf{Blt}_N = \perp$. Then we have that $\mathsf{VerifyTally}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y_1, \gamma_1) = \mathsf{OK}$ and by (A) $y_1 = \perp$, which is a contradiction.
   b) It is not the case that $\mathsf{Blt}_1 = \cdots = \mathsf{Blt}_N = \perp$. Then, by (B) we have that $\mathsf{VerifyTally}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y_0, \gamma_0) = \perp$, which contradicts the fact that $(y_0, \gamma_0)$ is accepted.
2) $y_0, y_1 \neq \perp$.
   Let $y_0, \gamma_0, y_1, \gamma_1$ be arbitrary strings in $\{0,1\}^\star \cup \{\perp\}$ such that $y_0, y_1 \neq \perp$. Suppose that $\mathsf{VerifyTally}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y_0, \gamma_0) = \mathsf{VerifyTally}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y_1, \gamma_1) = \mathsf{OK}$. The perfect soundness of $\mathsf{NIWI}^{\mathsf{dec}}$ implies that, for all $b \in \{0,1\}$, the proof $\gamma_b$ is computed on input some witness $(\mathcal{E}.\mathsf{Sk}'^b_1, \mathcal{E}.\mathsf{Sk}'^b_2, s^b_1, s^b_2, i^b_1, i^b_2)$.

In the following, when we talk about "condition X for proof $\gamma_b$", we refer to the items (1)-(5) in the definition of relation $\mathsf{R}^{\mathsf{dec}}$ in Fig. 6.

By the pigeon principle, there exists an index $i^\star \in [3]$ such that one of the following cases holds.

a) $i^\star = i_1^0 = i_2^1$. For all $b \in \{0,1\}$, let $(m_1^{i^\star,b}, \ldots, m_N^{i^\star,b})$ be the messages guaranteed by condition (3) of relation $\mathsf{R}^{\mathsf{dec}}$ for proof $\gamma_b$. Condition (1) for proof $\gamma_0$ (resp. $\gamma_1$) implies that the secret key $\mathsf{Sk}_1'^0$ (resp. $\mathsf{Sk}_2'^1$) is honestly computed and thus, the unique secret key property and the fact that it fulfills $\mathcal{E}.\mathsf{Pk}_{i_1^0} = \mathcal{E}.\mathsf{Pk}_{i^\star}$ (resp. $\mathcal{E}.\mathsf{Pk}_{i_2^1} = \mathcal{E}.\mathsf{Pk}_{i^\star}$) imply that for all $j \in [N]$, $\mathcal{E}.\mathsf{Decrypt}(\mathsf{Ct}_{j,i^\star}, \mathcal{E}.\mathsf{Sk}_1'^0) = \mathcal{E}.\mathsf{Decrypt}(\mathsf{Ct}_{j,i^\star}, \mathcal{E}.\mathsf{Sk}_2'^1)$.
Furthermore, condition (2) and (3) for proof $\gamma_0$ (resp. $\gamma_1$) imply that for all $j \in [N]$, either $m_j^{i^\star,0} = \perp$ or $m_j^{i^\star,0} = \mathcal{E}.\mathsf{Decrypt}(\mathsf{Ct}_{j,i^\star}, \mathcal{E}.\mathsf{Sk}_1') \in \mathcal{M}$ (resp. either $m_j^{i^\star,1} = \perp$ or $m_j^{i^\star,1} = \mathcal{E}.\mathsf{Decrypt}(\mathsf{Ct}_{j,i^\star}, \mathcal{E}.\mathsf{Sk}_2'^1) \in \mathcal{M}$).
Hence, for all $j \in [N]$, $m_j^{i^\star,0} = m_j^{i^\star,1} \in \mathcal{M} \cup \{\perp\}$.
Now, condition (4) for proof $\gamma_0$ (resp. $\gamma_1$) implies that either $y_0 = F(m_1^{i_1^0,0}, \ldots, m_N^{i_1^0,0})$ or $y_0 = \perp$ (resp. either $y_1 = F(m_1^{i_2^1,1}, \ldots, m_N^{i_2^1,1})$ or $y_1 = \perp$) and, as by hypothesis $y_0, y_1 \neq \perp$, it holds that $y_0 = y_1$.

b) $i^\star = i_2^0 = i_1^1$. This case is identical to the first one, except that we replace $i_1^0$ with $i_2^0$ and $i_2^1$ with $i_1^1$.

c) $i^\star = i_1^0 = i_1^1$. This case is identical to the first one, except that we replace $i_2^1$ with $i_1^1$.

d) $i^\star = i_2^0 = i_2^1$. This case is identical to the first one, except that we replace $i_1^0$ with $i_2^0$.

In all cases, if it is the case that $\mathsf{VerifyTally}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y_0, \gamma_0) = \mathsf{VerifyTally}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y_1, \gamma_1) = \mathsf{OK}$ then $y_0 = y_1$. In conclusion, the claim is proved.

From the previous claim, it follows that there exists a *unique* value $y^\star$ such that, for all $(y, \gamma)$ such that $y \neq \perp$, if $\mathsf{VerifyTally}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y, \gamma) = \mathsf{OK}$ then $y = y^\star$ (1). Moreover, it is easy to see that, for all $(y, \gamma)$, if $\mathsf{VerifyTally}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y, \gamma) = \mathsf{OK}$, there exist messages $m_1, \ldots, m_N \in \mathcal{M} \cup \{\perp\}$ such that $y = F(m_1, \ldots, m_N)$ (2).

Now, we have two mutually exclusive cases.

- For all $(y, \gamma)$ such that $y \neq \perp$, $\mathsf{VerifyTally}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y, \gamma) = \perp$. Then, letting $m_1, \ldots, m_N$ in the statement of the theorem be arbitrary messages in $\mathcal{M} \cup \{\perp\}$, the statement is verified with respect to $\mathsf{Pk}$ and $\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N$.
- There exists $(y', \gamma)$ such that $y' \neq \perp$ and $\mathsf{VerifyTally}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y', \gamma) = \mathsf{OK}$. In this case, (2) implies that there exist $m_1', \ldots, m_N' \in \mathcal{M} \cup \{\perp\}$ such that $y' = F(m_1', \ldots, m_N')$ (3). Hence, (1) and (3) together imply that $y^\star = F(m_1', \ldots, m_N')$ (4).
  Therefore, for all $(y, \gamma)$ such that $y \neq \perp$, if we have that $\mathsf{VerifyTally}(\mathsf{Pk}, \mathsf{Blt}_1, \mathsf{Blt}_N, y, \gamma) = \mathsf{OK}$ then (by (1)) $y = y^\star =$ (by (4)) $= F(m_1', \ldots, m_N')$.
  Then, for $m_1 \triangleq m_1', \ldots, m_N \triangleq m_N'$, the statement of condition (1) of verifiability is verified with respect to $\mathsf{Pk}$ and $\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N$.

In both cases, for $m_1 \triangleq m_1', \ldots, m_N \triangleq m_N'$, the statement of condition (1) of verifiability is verified with respect to $\mathsf{Pk}$ and $\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N$. As $\mathsf{Pk}$ and $\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N$ are arbitrary strings, the statement of condition (1) of verifiability is proven.

It is also easy to check that condition (2) of verifiability is satisfied. This follows straightforwardly from the perfect soundness of $\mathsf{NIWI}^{\mathsf{dec}}$. Thanks to $\mathsf{NIWI}^{\mathsf{dec}}$, the authority always proves that the public key of the PKE scheme is honestly generated. Therefore, by the perfect correctness of the PKE scheme, an honestly computed ballot for message $m$ for the $j$-th voter is decrypted to $m$ (because an honestly computed ballot, by definition, consists of three ciphertexts that encrypt the same message, and thus the value committed to in $Z$ is not relevant). Consequently, if the tally $y$ is different from $\perp$ (i.e., if the evaluation of the tally function is equal for all indices), then $y$ has to be compatible with $m$ at index $j$ (cf. Def. 2.1).

In essence, condition (2) is satisfied because the degree of freedom of the authority in creating a dishonest public key only allows it to set up the commitment dishonestly. This does not affect how honest ballots are decrypted and "counted". ∎

Note that, for the proof of the theorem above, the security of the commitment scheme $\mathsf{Com}$ is not needed, i.e., the theorem holds for any PPT algorithm $\mathsf{Com}$, even insecure ones.

## APPENDIX D
## PRIVACY OF OUR EVOTE

*Theorem D.1:* For all $N > 0$, all sets $\mathcal{M}, \Sigma \subset \{0,1\}^\star$, and all tally functions $F : (\mathcal{M} \cup \{\perp\})^N \to \Sigma \cup \{\perp\}$, if $\mathcal{E}$ is a perfectly correct PKE scheme with unique secret key (cf. Def. B.1), $\mathsf{Com}$ is a computationally hiding commitment scheme (cf. Def. B.2), and $\mathsf{NIWI}^{\mathsf{enc}}$ and $\mathsf{NIWI}^{\mathsf{dec}}$ are (one-message) NIWIs (cf. Def. B.4), respectively, for the relations $\mathsf{R}^{\mathsf{enc}}$ and $\mathsf{R}^{\mathsf{dec}}$, then $\mathsf{EVOTE}^{N,\mathcal{M},\Sigma,F,\mathcal{E},\mathsf{Com},\mathsf{NIWI}^{\mathsf{enc}},\mathsf{NIWI}^{\mathsf{dec}}}$ is $\mathsf{IND}$-Secure (cf. Def. 2.6).

**Proof:** Consider the following experiment $H_{\mathcal{A}}^Z(1^\lambda)$ between a challenger and a non-uniform PPT adversary $\mathcal{A}$ (henceforth, we often omit the parameters). $H^Z$ is equal to the experiment $\mathsf{Priv}_{\mathcal{A}}^{N,\mathcal{M},\Sigma,F,\mathsf{EVOTE}}$ except that the challenger sets the commitment $Z$ in the public key to be a commitment to 0 instead of 1. We define the output of the experiment to be a bit that is 1 if and only if all winning conditions are satisfied. Then, consider the following claim.

*Claim D.2:* The probability $P_0$ that $\mathcal{A}$ wins the experiment $\mathsf{Priv}_{\mathcal{A}}^{N,\mathcal{M},\Sigma,F,\mathsf{EVOTE}}$ is negligibly different from the probability $P_1$ that $\mathcal{A}$ wins game $H^Z$.

**Proof:** Suppose towards a contradiction that the difference between $P_0$ and $P_1$ is some non-negligible function $\epsilon(\lambda)$. We

construct an adversary $\mathcal{B}$ that breaks the computationally hiding property of $\mathsf{Com}$ with non-negligible probability.

$\mathcal{B}$ receives as input a commitment $\mathsf{com}$ that is either a commitment to 0 or to 1. For $l \in [3]$, $\mathcal{B}$ runs $\mathcal{E}.\mathsf{Setup}(1^\lambda)$ to compute $(\mathcal{E}.\mathsf{Pk}_l, \mathcal{E}.\mathsf{Sk}_l)$ and sets the public key $\mathsf{Pk} = (\mathcal{E}.\mathsf{Pk}_1, \ldots, \mathcal{E}.\mathsf{Pk}_3, Z = \mathsf{com})$. $\mathcal{B}$ follows the challenger of $\mathsf{Priv}_{\mathcal{A}}^{N,\mathcal{M},\Sigma,F,\mathsf{EVOTE}}$ to compute the remaining messages that are sent to the adversary. Finally, $\mathcal{B}$ gets the output $b'$ from $\mathcal{A}$. $\mathcal{B}$ outputs 1 if and only if all winning conditions are satisfied.

By hypothesis, if $\mathsf{com}$ is a commitment to 1, the probability that $\mathcal{B}$ outputs 1 equals the probability that $\mathcal{A}$ wins in $\mathsf{Priv}_{\mathcal{A}}^{N,\mathcal{M},\Sigma,F,\mathsf{EVOTE}}$, and if $\mathsf{com}$ is a commitment to 0, the probability that $\mathcal{B}$ outputs 1 equals the probability that $\mathcal{A}$ wins in $H^Z$. Thus, the advantage of $\mathcal{B}$ in breaking the computational hiding property of $\mathsf{Com}$ is $\epsilon(\lambda)$, which contradicts the assumption that the commitment scheme is computationally hiding. ▌

Before continuing with the proof, we would like to remark a subtle point. In the previous claim, we implicitly assumed that the adversary $\mathcal{B}$ is able to check all of the winning conditions efficiently. This is possible if $\mathcal{M}$ is efficiently enumerable and its cardinality, as well as the number of voters $N$, are *constant* in the security parameter. This could seem like resorting to "complexity leveraging" arguments. In fact, one could ask if our proof would break down if $N$ and $\mathcal{M}$ depend on the security parameter. However, the whole proof can be generalized to the case of $N$ and $|\mathcal{M}|$ polynomial in the security parameter by using the following observation. Let $A$ be the event that $\mathcal{A}$ submits challenges that satisfy the winning condition. Then, if the probability that $\mathcal{A}$ wins the $\mathsf{Priv}_{\mathcal{A}}^{N,\mathcal{M},\Sigma,F,\mathsf{EVOTE}}$ is non-negligible, then the event $A$ must occur with non-negligible probability and, conditioned on it, $\mathcal{A}$ wins with non-negligible probability as well. Therefore, the rest of the proof would follow analyzing the probability that $\mathcal{A}$ wins in the next hybrid experiments conditioned under the occurrence of the event that, in such experiments, $\mathcal{A}$ submit challenges satisfying the winning condition. As we will see now, a similar "conditioning" argument will be anyhow necessary for the rest of the proof.

Let $E^1$ be the event that, in experiment $H^Z$, $\mathcal{A}$ submits as challenge two tuples $M_0 = (m_{0,1}, \ldots, m_{0,N})$ and $M_1 = (m_{1,1}, \ldots, m_{1,N})$ and a set $S \subset [N]$ that fulfill the following condition: there exists $j \in S$ such that $m_{0,j} = m_{1,j}$ and, letting $\mathsf{Blt}_j = m_{0,j} = (\mathsf{Ct}_{j,1}, \ldots, \mathsf{Ct}_{j,3}, \pi_j)$ (suppose that $m_{0,j}$ can be parsed that way), it holds that $\mathsf{VerifyBallot}(\mathsf{Pk}, j, \mathsf{Blt}_j) = \mathsf{OK}$ but there exist $i_1, i_2 \in [3], i_1 \neq i_2$ such that $\mathcal{E}.\mathsf{Decrypt}(\mathsf{Ct}_{i_1}, \mathsf{Sk}_{i_1}) \neq \mathcal{E}.\mathsf{Decrypt}(\mathsf{Ct}_{i_2}, \mathsf{Sk}_{i_2})$.

*Claim D.3:* The probability that $E^1$ occurs is negligible.

**Proof:** Suppose towards a contradiction that the probability of occurrence of $E^1$ be some non-negligible function $\epsilon(\lambda)$. We construct an adversary $\mathcal{B}$ that breaks the computationally hiding property of $\mathsf{Com}$ with non-negligible probability.

$\mathcal{B}$ receives as input a commitment $\mathsf{com}$ that is either a commitment to 0 or to 1. For $l \in [3]$, $\mathcal{B}$ runs $\mathcal{E}.\mathsf{Setup}(1^\lambda)$ to compute $(\mathcal{E}.\mathsf{Pk}_l, \mathcal{E}.\mathsf{Sk}_l)$ and sets the public key $\mathsf{Pk} = (\mathcal{E}.\mathsf{Pk}_1, \ldots, \mathcal{E}.\mathsf{Pk}_3, Z = \mathsf{com})$. $\mathcal{B}$ follows the challenger of $\mathsf{Priv}_{\mathcal{A}}^{N,\mathcal{M},\Sigma,F,\mathsf{EVOTE}}$ to compute the remaining messages that are sent to the adversary. $\mathcal{B}$ receives two tuples $M_0 = (m_{0,1}, \ldots, m_{0,N})$ and $M_1 = (m_{1,1}, \ldots, m_{1,N})$ and a set $S \subset [N]$ from the adversary.

For all $j \in S$, $\mathcal{B}$ checks whether the following conditions are all satisfied: $m_{0,j} = m_{1,j}$ and, after setting $\mathsf{Blt}_j = m_{0,j}$, $\mathsf{Blt}_j$ can be parsed as $(\mathsf{Ct}_{j,1}, \ldots, \mathsf{Ct}_{j,3}, \pi_j)$ and it holds that $\mathsf{VerifyBallot}(\mathsf{Pk}, j, \mathsf{Blt}_j) = \mathsf{OK}$ but there exist $i_1, i_2 \in [3], i_1 \neq i_2$ such that $\mathcal{E}.\mathsf{Decrypt}(\mathsf{Ct}_{i_1}, \mathsf{Sk}_{i_1}) \neq \mathcal{E}.\mathsf{Decrypt}(\mathsf{Ct}_{i_2}, \mathsf{Sk}_{i_2})$. If for some $j \in S$ the conditions are satisfied, $\mathcal{B}$ outputs 0, otherwise it outputs 1.

If $\mathsf{com}$ is a commitment to 1, the perfect soundness of $\mathsf{NIWI}^{\mathsf{enc}}$ and the definition of relation $\mathsf{R}^{\mathsf{enc}}$ guarantee that the conditions above are never satisfied for any $j \in S$. Therefore, if $\mathsf{com}$ is a commitment to 1 $\mathcal{B}$ outputs 1 with probability 1.

On the other hand, if $\mathsf{com}$ is a commitment to 0, the probability that the conditions are satisfied for some $j \in [S]$ equals the probability of $E^1$. Therfore, $\mathcal{B}$ outputs 0 with probability $\epsilon$ and 1 with probability $1 - \epsilon$. In conclusion, the advantage of $\mathcal{B}$ in breaking the computationally hiding property of $\mathsf{Com}$ is $\epsilon(\lambda)$, which contradicts the assumption that the commitment scheme is computationally hiding. ▌

From Claim D.2 and Claim D.3, we now know that, for some negligible function $\mathsf{negl}(\cdot)$, the following equations hold:

$$\left|\Pr\left[\,\mathsf{Priv} = 1\,\right] - \Pr\left[\,H^Z = 1\,\right]\right| \leq \mathsf{negl}(\lambda), \tag{1}$$

$$\Pr\left[\,E^1\,\right] \leq \mathsf{negl}(\lambda), \tag{2}$$

$$\begin{aligned}\Pr\left[\,H^Z = 1\,\right] = \Pr\left[\,H^Z = 1 | E^1\,\right]\Pr\left[\,E^1\,\right] + \\ \Pr\left[\,H^Z = 1 | \bar{E}^1\,\right]\Pr\left[\,\bar{E}^1\,\right] \leq \\ \mathsf{negl} + \Pr\left[\,H^Z = 1 | \bar{E}^1\,\right](1 - \mathsf{negl}).\end{aligned} \tag{3}$$

Here and henceforth, we omit the parameters, but it is meant that the experiments are parameterized by $\lambda$ and $\mathsf{negl}(\cdot)$.

Thus, to show that $\Pr\left[\,\mathsf{Priv} = 1\,\right]$ equals $1/2$ plus a negligible quantity, it is sufficient to show that $\Pr\left[\,H^Z = 1 | \bar{E}^1\,\right]$ equals $1/2$ plus a negligible quantity. We prove the latter by means of a series of hybrid experiments. In the following we analyze the behavior of the adversary conditioned on the occurrence of the event $\bar{E}^1$.

- **Hybrid** $H_1$**.** Experiment $H_1$ is equal to the experiment $H^Z$ except that the challenger sets $b = 0$.
- **Hybrid** $H_2^k$, for $k = 0, \ldots, N$. For all $k = 0, \ldots, N$, experiment $H_2^k$ is identical to experiment $H_1$ except that, for all $j = 1, \ldots, k$ such that $j \notin S$, the challenger computes $\mathsf{Ct}_{k,3}$ on input $m_{1,k}$. Note that $H_2^0$ is identical to $H_1$.

*Claim D.4:* $\left|\Pr\left[H_2^{k-1}=1|\bar{E}^1\right]-\Pr\left[H_2^{k}=1|\bar{E}^1\right]\right|$ is negligible for all $k=1,\ldots,N$,.

**Proof:** Suppose toward a contradiction that the difference between such probabilities is a non-negligible function $\epsilon(\lambda)$. We construct an adversary $\mathcal{B}$ that has advantage at most $\epsilon(\lambda)$ against the IND-CPA security of $\mathcal{E}$.

$\mathcal{B}$ receives from the challenger of IND-CPA a public key $\mathsf{pk}$ and sets $\mathsf{Pk}_3=\mathsf{pk}$. For $l\in[2]$, $\mathcal{B}$ runs $\mathcal{E}.\mathsf{Setup}$ to compute $(\mathcal{E}.\mathsf{Pk}_l,\mathcal{E}.\mathsf{Sk}_l)$, computes $Z\leftarrow\mathsf{Com}(0)$ and runs $\mathcal{A}$ on input $\mathsf{Pk}=(\mathcal{E}.\mathsf{Pk}_1,\mathcal{E}.\mathsf{Pk}_2,\mathcal{E}.\mathsf{Pk}_3,Z)$.

$\mathcal{A}$ outputs two tuples $(m_{0,1},\ldots,m_{0,N})$ and $(m_{1,1},\ldots,m_{1,N})$ and a set $S$. If $k\in S$, $\mathcal{B}$ sends $(0,0)$ as its pair of challenge messages to the IND-CPA challenger, which returns the challenge ciphertext $\mathsf{ct}^\star$ to $\mathcal{B}$. If $k\notin S$, $\mathcal{B}$ sends $(m_{0,k},m_{1,k})$ as its pair of challenge messages to the IND-CPA challenger, which returns the challenge ciphertext $\mathsf{ct}^\star$ to $\mathcal{B}$.

If $k\in S$, $\mathcal{B}$ sets $\mathsf{Blt}_j$ as the challenger in the real experiment would do, else $\mathcal{B}$ computes $\mathsf{Blt}_k=(\mathsf{Ct}_{k,1},\mathsf{Ct}_{k,2},\mathsf{ct}^\star)$ by computing $\mathsf{Ct}_{k,1}$ and $\mathsf{Ct}_{k,2}$ on input $m_{0,j}$. For all $j\in[N](j\neq k)$, $\mathcal{B}$ computes the ballots $\mathsf{Blt}_j$ exactly as the challenger in both experiments would do. $\mathcal{B}$ computes $y$ using $\mathsf{EvalTally}$ and uses the 2 secret keys $\mathcal{E}.\mathsf{Sk}_1,\mathcal{E}.\mathsf{Sk}_2$ to compute a proof $\gamma$ exactly as the challenger in both experiments would do. $\mathcal{B}$ sends $\mathcal{A}$ the computed ballots along with $(y,\gamma)$ and returns the output of $\mathcal{A}$.

It is easy to see that, if $\mathsf{ct}^\star$ is an encryption of $m_{0,k}$ and if $k\notin S$, then $\mathcal{B}$ simulates experiment $H_2^{k-1}$ and if $\mathsf{ct}^\star$ is an encryption of $m_{1,k}$ and $k\notin S$, then $\mathcal{B}$ simulates experiment $H_2^{k}$. If $k\in S$ the advantage of $\mathcal{A}$ is 0.

Therefore, $\mathcal{B}$ has non-negligible probability of winning the IND-CPA game, which contradicts the assumption that the PKE scheme fulfills the IND-CPA property. ▮

- **Hybrid** $H_3$. Experiment $H_3$ is identical to experiment $H_2^N$ except that the challenger computes the proof $\gamma$ on input a witness that contains indices $(1,3)$ and secret keys $(\mathsf{Sk}_1,\mathsf{Sk}_3)$ (precisely, the witness contains the randomness used to compute those secret keys, but henceforth, for simplicity, we omit this detail).
*Claim D.5:* $\left|\Pr\left[H_2^{N}=1|\bar{E}^1\right]-\Pr\left[H_3=1|\bar{E}^1\right]\right|$ is negligible.

  **Proof:** The proof follows from the WI property of $\mathsf{NIWI}^{\mathsf{dec}}$. We observe that both the randomness used to compute $(\mathsf{Sk}_1,\mathsf{Sk}_2)$ and the randomness used to compute $(\mathsf{Sk}_1,\mathsf{Sk}_3)$ constitute valid witnesses for the statement $(\mathsf{Blt}_1,\ldots,\mathsf{Blt}_N,\mathcal{E}.\mathsf{Pk}_1,\ldots,\mathcal{E}.\mathsf{Pk}_3,y)$. Additionally, we observe that, if event $\bar{E}^1$ occurs, any ballot in the set $S$ is in both experiments either replaced by $\perp$, if $\mathsf{VerifyBallot}$ refuses it, or decrypted to the same value. Consequently, the tally is identical in both experiments. ▮

- **Hybrid** $H_4^k$, for $k=0,\ldots,N$. For all $k=0,\ldots,N$, experiment $H_4^k$ is identical to experiment $H_3$ except that, for all $j=1,\ldots,k$ such that $j\notin S$, the challenger computes $\mathsf{Ct}_{k,2}$ on input $m_{1,k}$. Note that $H_4^0$ is identical to $H_3$.
*Claim D.6:* For all $k=1,\ldots,N$, we have that $\left|\Pr\left[H_4^{k-1}=1|\bar{E}^1\right]-\Pr\left[H_4^{k}=1|\bar{E}^1\right]\right|$ is negligible.

  **Proof:** The proof is identical to the one for Claim D.4 except that the third index and the second index are swapped. ▮

- **Hybrid** $H_5$. Experiment $H_5$ is identical to experiment $H_4^N$ except that the challenger computes the proof $\gamma$ on input a witness that contains indices $(2,3)$ and secret keys $(\mathsf{Sk}_2,\mathsf{Sk}_3)$.
*Claim D.7:* $\left|\Pr\left[H_4^{N}=1|\bar{E}^1\right]-\Pr\left[H_5=1|\bar{E}^1\right]\right|$ is negligible.

  **Proof:** This follows straightforwardly from the WI property of $\mathsf{NIWI}^{\mathsf{dec}}$. We observe that both the randomness used to compute $(\mathsf{Sk}_1,\mathsf{Sk}_3)$ and the randomness used to compute $(\mathsf{Sk}_2,\mathsf{Sk}_3)$ constitute valid witnesses for the statement $(\mathsf{Blt}_1,\ldots,\mathsf{Blt}_N,\mathcal{E}.\mathsf{Pk}_1,\ldots,\mathcal{E}.\mathsf{Pk}_3,y)$. Additionally, we observe that, if event $\bar{E}^1$ occurs, any ballot in the set $S$ is in both experiments either replaced by $\perp$, if $\mathsf{VerifyBallot}$ refuses it, or decrypted to the same value. Consequently, the tally is identical in both experiments. ▮

- **Hybrid** $H_6^k$, for $k=0,\ldots,N$. For all $k=0,\ldots,N$, experiment $H_6^k$ is identical to experiment $H_5$ except that, for all $j=1,\ldots,k$ such that $j\notin S$, the challenger computes $\mathsf{Ct}_{k,1}$ on input $m_{1,k}$. Note that $H_6^0$ is identical to $H_5$.
*Claim D.8:* For all $k=1,\ldots,N$, we have that $\left|\Pr\left[H_6^{k-1}=1|\bar{E}^1\right]-\Pr\left[H_6^{k}=1|\bar{E}^1\right]\right|$ is negligible.

  **Proof:** The proof is identical to the one for Claim D.4 except that the third index and the first index are swapped. ▮

- **Hybrid** $H_7$. Experiment $H_7$ is identical to experiment $H_6^N$ except that the challenger sets $b=1$ (so that the winning condition be computed differently) and computes the proof $\gamma$ on input a witness that contains indices $(1,2)$ and secret keys $(\mathsf{Sk}_1,\mathsf{Sk}_2)$.
*Claim D.9:* $\left|\Pr\left[H_6^{N}=1|\bar{E}^1\right]-\Pr\left[H_7=0|\bar{E}^1\right]\right|$ is negligible.

  **Proof:** The proof follows straightforwardly from the WI property of $\mathsf{NIWI}^{\mathsf{dec}}$. We observe that both the randomness used to compute $(\mathsf{Sk}_1,\mathsf{Sk}_2)$ and the randomness used to compute $(\mathsf{Sk}_2,\mathsf{Sk}_3)$ constitute valid witnesses for the statement $(\mathsf{Blt}_1,\ldots,\mathsf{Blt}_N,\mathcal{E}.\mathsf{Pk}_1,\ldots,\mathcal{E}.\mathsf{Pk}_3,y)$. Additionally, we observe that, if event $\bar{E}^1$ occurs, any ballot in the set $S$ is in both experiments either replaced by $\perp$, if $\mathsf{VerifyBallot}$ refuses it, or decrypted to the same value. Consequently, the tally is identical in both experiments.

  Note that according to the proof received, an adversary against NIWI can emulate experiment $H_6^N$ or $H_7$, and

return the output of $\mathcal{A}$. In the first case, the probability that $\mathcal{A}$ outputs 0 is exactly $\Pr\left[H_6^N = 1|\bar{E}^1\right]$ because the winning condition is computed with respect to $b = 0$, whereas in the second case it is $\Pr\left[H_7 = 0|\bar{E}^1\right]$ because the winning condition is computed with respect to $b = 1$. ∎

Now, consider Equation 4 in Fig. 7. Claim D.2 and equations 2, 3 and 4 imply that $\Pr[\mathsf{Priv} = 1] \leq 1/2 + \nu$ for some negligible function $\nu$. Therefore, the advantage of the adversary is negligible and the theorem is proven. ∎

*Corollary D.10:* If the Decision Linear assumption (see Section B) holds, then there exists a private and verifiable eVote.

**Proof:** Boneh *et al.* [41] show the existence of a PKE with perfect correctness and unique secret key that fulfills the IND-CPA property under the Decision Linear assumption. Groth *et al.* [28] show the existence of (one-message) NIWI (with perfect soundness) for all languages in NP and of statistically binding commitments. Both constructions are secure under the Decision Linear assumption. Then, because Theorem C.1 and Theorem D.1 are proven, the corollary follows. ∎

## APPENDIX E
## INSTANTIATION OF EVOTE

In this section, we describe an instantiation of our e-voting scheme EVOTE. In Section E-A, we describe the algorithms of our efficient instantiation without describing the details of the one-message NIWI proofs for the relations $\mathsf{R}^{\mathsf{enc}}$ and $\mathsf{R}^{\mathsf{dec}}$. The next subsections describe how we construct efficient one-message NIWI proofs for the relations $\mathsf{R}^{\mathsf{enc}}$ and $\mathsf{R}^{\mathsf{dec}}$. First, in Section E-B, we define NIWI proofs that are secure under a trusted setup, which are used as building block of our construction for one-message NIWI. In Section E-C, we summarize Groth-Sahai NIWI proofs, which provide us with NIWI proofs secure under a trusted setup for the satisfiability of equations over bilinear groups. In Section E-D, we describe our construction for one-message NIWI. When instantiated with Groth-Sahai proofs, we obtain a one-message NIWI for the satisfiability of equations over bilinear groups. We analyze the efficiency of Groth-Sahai NIWI proofs in Section E-E. In Section E-F, we describe a concrete instantiation of the relations for $\mathsf{R}^{\mathsf{enc}}$ and $\mathsf{R}^{\mathsf{dec}}$ over bilinear groups, which can be used in our construction in Section E-A. Finally, in Section E-G, we analyze the efficiency of our instantiation of EVOTE.

### A. Algorithms of our Efficient Instantiation of EVOTE

EVOTE uses a public-key encryption scheme with algorithms $\mathcal{E} = (\mathcal{E}.\mathsf{Setup}, \mathcal{E}.\mathsf{Encrypt}, \mathcal{E}.\mathsf{Decrypt})$ with perfect correctness and unique secret key (see Def. B.1). We use the DLIN encryption scheme, which we recall in Section B, in order to instantiate the public-key encryption scheme.

EVOTE also uses a perfectly binding commitment scheme Com. We also use the DLIN encryption scheme for this purpose. A DLIN ciphertext is used as a perfectly binding commitment.

Finally, EVOTE uses two one-message NIWI proofs for the relations $\mathsf{R}^{\mathsf{enc}}$ and $\mathsf{R}^{\mathsf{dec}}$. We describe in the next sections how to compute those NIWI proofs.

We use as tally function $F = \sum_{j=1}^{N} v_j$, i.e., the sum of the votes $v_j \in \{0, 1\}$. We represent $\{0, 1\}$ as $1, g \in \mathbb{G}$.

In Fig. 8, we describe the algorithms of our instantiation of EVOTE. In our scheme in Section IV, the setup algorithm consists of 3 public keys and a commitment value $Z$. In this instantiation, the setup algorithm outputs an additional public key $\mathsf{Pk}_4$ to be used as the parameters of the commitment scheme used for computing $Z$.

### B. NIWI Proofs Secure Under a Trusted Setup

The following definitions are taken from [32]. Let $R$ be a polynomial time computable binary relation. For tuples $(gk, x, w) \in R$, we call $gk$ the public parameter, $x$ the instance and $w$ the witness. Let $L$ be the NP-language consisting of the instances $x$ for which witnesses $w$ exist such that $(gk, x, w) \in R$.

*Definition E.1:* [NIWI proof system in the CRS model] A NIWI proof system in the CRS model for the relation $R$ consists of four algorithms NIWISetup, NIWIKeygen, NIWIProve and NIWIVerify. On input a security parameter $1^k$, $\mathsf{NIWISetup}(1^k)$ outputs a setup $(gk, sk)$ consisting of the public parameter $gk$ and the secret parameter $sk$. (For example, in Groth-Sahai proofs, $gk$ is a public parameter that represents the description of a pairing group setup.) $\mathsf{NIWIKeygen}(gk, sk)$ outputs a common reference string $crs$. $\mathsf{NIWIProve}(gk, crs, w, x)$ checks whether $(gk, x, w) \in R$ and if so outputs a proof $\pi$. $\mathsf{NIWIVerify}(gk, crs, x, \pi)$ outputs 1 if $\pi$ is a valid proof that $x \in L$, or 0 if that is not the case. A NIWI system must fulfill the following properties.

- Completeness: Completeness requires that algorithm NIWIVerify accepts the proofs computed by algorithm NIWIProve. More formally, for all $(gk, w, x) \in R$, the completeness property is defined as follows:

$$\Pr\left[\begin{array}{l}(gk, sk) \leftarrow \mathsf{NIWISetup}(1^k);\\ crs \leftarrow \mathsf{NIWIKeygen}(gk, sk);\\ \pi \leftarrow \mathsf{NIWIProve}(gk, crs, w, x) :\\ 1 = \mathsf{NIWIVerify}(gk, crs, x, \pi)\end{array}\right] = 1.$$

- Perfect Soundness: For every non-uniform adversary $\mathcal{A}$, it holds that:

$$\Pr\left[\begin{array}{l}(gk, sk) \leftarrow \mathsf{NIWISetup}(1^k);\\ crs \leftarrow \mathsf{NIWIKeygen}(gk, sk);\\ (x, \pi) \leftarrow \mathcal{A}(gk, crs) :\\ \mathsf{NIWIVerify}(gk, crs, x, \pi) = 0 \wedge x \notin L\end{array}\right] = 1.$$

  Computational soundness holds against any non-uniform PPT adversary $\mathcal{A}$.
- Composable witness indistinguishability: The standard definition of witness indistinguishability requires that proofs computed on input different witnesses for the same instance are computationally indistinguishable. Composable witness indistinguishability also requires that there is a simulator $\mathcal{S}$ that generates a simulated common

$$
\begin{aligned}
&\Pr\left[H^Z = 1|\bar{E}^1\right] = \\
&\Pr\left[H^Z = 1|\bar{E}^1 \wedge b = 0\right]\Pr[\,b = 0\,] + \Pr\left[H^Z = 1|\bar{E}^1 \wedge b = 1\right]\Pr[\,b = 1\,] = \\
&= 1/2 \cdot \left(\Pr\left[H^Z = 1|\bar{E}^1 \wedge b = 0\right] + \Pr\left[H^Z = 1|\bar{E}^1 \wedge b = 1\right]\right) = \\
&\text{(since } H_1 \text{ is identically distributed to } H^Z \text{ with bit } b = 0 \text{ and } H_7 \text{ to } H^Z \text{ with } b = 1) \\
&= 1/2 \cdot \left(\Pr\left[H_1 = 1|\bar{E}^1\right] + \Pr\left[H_7 = 1|\bar{E}^1\right]\right) = \\
&= 1/2 + 1/2 \cdot \left(\Pr\left[H_1 = 1|\bar{E}^1\right] - \Pr\left[H_7 = 0|\bar{E}^1\right]\right) = \\
&\text{(since } H_1 \text{ (resp. } H_3\text{, } H_5\text{) is identically distributed to } H_2^0 \text{ (resp. } H_4^0\text{, } H_6^0\text{))} \\
&= 1/2 + 1/2 \cdot (\sum_{k=0}^{N-1}(\Pr\left[H_2^k = 1|\bar{E}^1\right] - \Pr\left[H_2^{k+1} = 1|\bar{E}^1\right]) + (\Pr\left[H_2^N = 1|\bar{E}^1\right] - \Pr\left[H_4^0 = 1|\bar{E}^1\right]) + \\
&\quad \sum_{k=0}^{N-1}(\Pr\left[H_4^k = 1|\bar{E}^1\right] - \Pr\left[H_4^{k+1} = 1|\bar{E}^1\right]) + (\Pr\left[H_4^N = 1|\bar{E}^1\right] - \Pr\left[H_6^0 = 1|\bar{E}^1\right]) + \\
&\quad \sum_{k=0}^{N-1}(\Pr\left[H_6^k = 1|\bar{E}^1\right] - \Pr\left[H_6^{k+1} = 1|\bar{E}^1\right]) + (\Pr\left[H_6^N = 1|\bar{E}^1\right] - \Pr\left[H_7 = 0|\bar{E}^1\right])) \leq \\
&\leq 1/2 + 1/2 \cdot |(\sum_{k=0}^{N-1}(\Pr\left[H_2^k = 1|\bar{E}^1\right] - \Pr\left[H_2^{k+1} = 1|\bar{E}^1\right]) + (\Pr\left[H_2^N = 1|\bar{E}^1\right] - \Pr\left[H_4^0 = 1|\bar{E}^1\right]) + \\
&\quad \sum_{k=0}^{N-1}(\Pr\left[H_4^k = 1|\bar{E}^1\right] - \Pr\left[H_4^{k+1} = 1|\bar{E}^1\right]) + (\Pr\left[H_4^N = 1|\bar{E}^1\right] - \Pr\left[H_6^0 = 1|\bar{E}^1\right]) + \\
&\quad \sum_{k=0}^{N-1}(\Pr\left[H_6^k = 1|\bar{E}^1\right] - \Pr\left[H_6^{k+1} = 1|\bar{E}^1\right]) + (\Pr\left[H_6^N = 1|\bar{E}^1\right] - \Pr\left[H_7 = 0|\bar{E}^1\right]))| \leq \\
&\text{(by the triangle inequality)} \\
&\leq 1/2 + 1/2 \cdot (\sum_{k=0}^{N-1}|\Pr\left[H_2^k = 1|\bar{E}^1\right] - \Pr\left[H_2^{k+1} = 1|\bar{E}^1\right]| + |(\Pr\left[H_2^N = 1|\bar{E}^1\right] - \Pr\left[H_4^0 = 1|\bar{E}^1\right])| + \\
&\quad \sum_{k=0}^{N-1}|\Pr\left[H_4^k = 1|\bar{E}^1\right] - \Pr\left[H_4^{k+1} = 1|\bar{E}^1\right]| + |\Pr\left[H_4^N = 1|\bar{E}^1\right] - \Pr\left[H_6^0 = 1|\bar{E}^1\right]| + \\
&\quad \sum_{k=0}^{N-1}|\Pr\left[H_6^k = 1|\bar{E}^1\right] - \Pr\left[H_6^{k+1} = 1|\bar{E}^1\right]| + |\Pr\left[H_6^N = 1|\bar{E}^1\right] - \Pr\left[H_7 = 0|\bar{E}^1\right]|) \leq \\
&\text{(by Claims D.4 - D.9)} \\
&\leq 1/2 + 1/2 \cdot 3k \cdot \mathsf{negl}\text{, where } \mathsf{negl} \text{ is the sum of the negligible functions guaranteed by Claims D.4 - D.9.}
\end{aligned}
\tag{4}
$$

Fig. 7. Equation 4

reference string that is indistinguishable from a real one. Additionally, on a simulated common reference string there is no information to distinguish witnesses that might have been used to construct the proof. More formally, there exists a PPT simulator $\mathcal{S}$ such that for all non-uniform PPT adversaries $\mathcal{A}$, it holds that:

$$
\left|\Pr\left[\begin{array}{l}(gk, sk) \leftarrow \mathsf{NIWISetup}(1^k);\\ crs \leftarrow \mathsf{NIWIKeygen}(gk, sk):\\ \mathcal{A}(gk, crs) = 1\end{array}\right] -
\Pr\left[\begin{array}{l}(gk, sk) \leftarrow \mathsf{NIWISetup}(1^k);\\ crs \leftarrow \mathcal{S}(gk, sk):\\ \mathcal{A}(gk, crs) = 1\end{array}\right]\right| \in \mathsf{negl}(k).
$$

Moreover, for all non-uniform adversaries $\mathcal{A}$, it holds that:

$$
\left|\Pr\left[\begin{array}{l}(gk, sk) \leftarrow \mathsf{NIWISetup}(1^k);\\ crs \leftarrow \mathcal{S}(gk, sk);\\ (x, w_0, w_1) \leftarrow \mathcal{A}(gk, crs);\\ \pi \leftarrow \mathsf{NIWIProve}(gk, crs, x, w_0):\\ \mathcal{A}(\pi) = 1 \ \wedge (gk, x, w_0) \in R\end{array}\right] -
\Pr\left[\begin{array}{l}(gk, sk) \leftarrow \mathsf{NIWISetup}(1^k);\\ crs \leftarrow \mathcal{S}(gk, sk);\\ (x, w_0, w_1) \leftarrow \mathcal{A}(gk, crs);\\ \pi \leftarrow \mathsf{NIWIProve}(gk, crs, x, w_1):\\ \mathcal{A}(\pi) = 1 \ \wedge (gk, x, w_1) \in R\end{array}\right]\right| = 0.
$$

### C. Groth-Sahai NIWI Proofs

Groth and Sahai [32] show how to compute NIWI proofs under a trusted setup (i.e., in the CRS model) for satisfiability of equations over bilinear groups. The Groth-Sahai non-interactive system can be instantiated in two settings (based on the DLIN assumption): in the "binding" setting, it fulfills the perfect soundness and composable witness indistinguishability properties, whereas in the "hiding" setting it fulfills computational soundness and composable witness indistinguishability (actually more).

Let us describe the instantiation of Groth-Sahai NIWI proofs in the CRS model based on the DLIN assumption in both

- $\mathsf{Setup}(1^\lambda)$: on input the security parameter in unary, do the following.
  1) Compute a bilinear map setup $\Gamma = (p, \mathbb{G}, \mathbb{G}_t, \mathbf{e}, g) \leftarrow \mathcal{G}(1^\lambda)$.
  2) Compute four key pairs for the DLIN encryption scheme. For $l = 1$ to $4$, run $(\mathcal{E}.\mathsf{Pk}_l, \mathcal{E}.\mathsf{Sk}_l) \leftarrow \mathcal{E}.\mathsf{Setup}(\Gamma)$, where $\mathcal{E}.\mathsf{Sk}_l = (g_l, f_l, h_l)$ and $(\mathcal{E}.\mathsf{Sk}_l) = (x_l, y_l)$.
  3) Compute a perfectly binding commitment $Z$ to 1. We use a DLIN encryption of 1, i.e., we run $Z \leftarrow \mathcal{E}.\mathsf{Encrypt}(\mathcal{E}.\mathsf{Pk}_4, g)$, where $Z = (a, b, c)$. Note that we represent 1 as $g \in \mathbb{G}$.
  4) Output $\mathsf{Pk} \leftarrow (\Gamma, \mathcal{E}.\mathsf{Pk}_1, \ldots, \mathcal{E}.\mathsf{Pk}_4, Z)$ and $\mathsf{Sk} \leftarrow (\mathcal{E}.\mathsf{Sk}_1, \mathcal{E}.\mathsf{Sk}_2)$.
- $\mathsf{Cast}(\mathsf{Pk}, j, v)$: on input the public key $\mathsf{Pk}$, the voter index $j \in [N]$, and a vote $v$, do the following.
  1) Check that $v$ is in $\{1, g\} \in \mathbb{G}$. We remark that, in this instantiation example, we do not consider the possibility of encrypting $\perp$.
  2) For all $l \in [3]$, compute $\mathsf{Ct}_{j,l} = \mathsf{Encrypt}(\mathcal{E}.\mathsf{Pk}_l, v)$, where $\mathsf{Ct}_{j,l} = (a_{j,l}, b_{j,l}, c_{j,l})$.
  3) Compute a one-message NIWI $\pi_j$ for the relation $\mathsf{R}^{\mathsf{enc}}$. We describe a construction for one-message NIWI in Section E-D and we describe an instantiation of the relation $\mathsf{R}^{\mathsf{enc}}$ over bilinear groups in Section E-F.
  4) Output $\mathsf{Blt}_j = (\mathsf{Ct}_{j,1}, \ldots, \mathsf{Ct}_{j,3}, \pi_j)$.
- $\mathsf{VerifyBallot}(\mathsf{Pk}, j, \mathsf{Blt})$: on input the public key $\mathsf{Pk}$, the voter index $j \in [N]$, and a ballot $\mathsf{Blt}$, verify the proof $\pi_j$ in the ballot by using the verification algorithm of our one-message NIWI construction in Section E-D for the relation $\mathsf{R}^{\mathsf{enc}}$ over bilinear groups in Section E-F. Output the verification result.
- $\mathsf{EvalTally}(\mathsf{Pk}, \mathsf{Sk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N)$: on input the public key $\mathsf{Pk}$, the secret key $\mathsf{Sk}$, and $N$ strings $(\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N)$ that can be either ballots cast by a voter or the special symbol $\perp$, do the following.
  1) For all $j \in [N]$, if $\mathsf{VerifyBallot}(\mathsf{Pk}, j, \mathsf{Blt}_j) = \perp$, set $\mathsf{Blt}_j = \perp$. If, for all $j \in [N]$, $\mathsf{Blt}_j = \perp$, then output $(y = \perp, \gamma = \perp)$.
  2) For all $j \in [N], l \in [2]$, set $m_{j,l} \leftarrow \mathcal{E}.\mathsf{Decrypt}(\mathsf{Ct}_{j,l}, \mathcal{E}.\mathsf{Sk}_l)$. Recall that in this instantiation example, we do not consider the possibility of encrypting $\perp$.
  3) For all $l \in [2]$, compute $y_l = \prod_{j=1}^{N} m_{j,l}$. Recall that we encode the votes $\{0, 1\}$ in the exponent and thus, for each $l \in [2]$, $y_l$ should correspond to $g^{\sum v_j}$, where the sum is over the voters whose ballots passed the verification ballot test. $\sum v_j$ can be computed (by brute force) by doing $\mathbf{dlog}_g y_l$.
  4) If $y_1 = y_2$ then set $y = y_1$, else set $y = \perp$.
  5) Compute a one-message NIWI $\gamma$ for the relation $\mathsf{R}^{\mathsf{dec}}$. We describe a construction for one-message NIWI in Section E-D and we describe an instantiation of the relation $\mathsf{R}^{\mathsf{dec}}$ over bilinear groups in Section E-F.
  6) Output $(y, \gamma)$.
- $\mathsf{VerifyTally}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y, \gamma)$: on input the public key $\mathsf{Pk} = (\Gamma, \mathcal{E}.\mathsf{Pk}_1, \ldots, \mathcal{E}.\mathsf{Pk}_4, Z)$, $N$ strings that can be either ballots cast by a voter or the special symbol $\perp$, a tally $y$ and a proof $\gamma$ of tally correctness, do the following.
  1) If there exist two indices $i_1, i_2 \in [3]$ such that $i_1 \neq i_2$ and $\mathcal{E}.\mathsf{Pk}_{i_1} = \mathcal{E}.\mathsf{Pk}_{i_2}$ return $\perp$. This step is necessary because in our instantiation of $\mathsf{R}^{\mathsf{dec}}$ we do not directly enforce that the two secret keys given as witness for the relation correspond to different indices. However, our equations are satisfied if and only if both of the two secret keys correspond to one of the public keys $\mathcal{E}.\mathsf{Pk}_1, \mathcal{E}.\mathsf{Pk}_2, \mathcal{E}.\mathsf{Pk}_3$ and thus, as DLIN encryption enjoys the property of unique secret key, verifying that there are no two equal public keys guarantees that the two secret keys that satisfy the relation are for two different indices.
  2) For all $j \in [N]$, if $\mathsf{VerifyBallot}(\mathsf{Pk}, j, \mathsf{Blt}_j) = \perp$, set $\mathsf{Blt}_j = \perp$.
  3) If $y \neq \perp$, then verify the proof $\gamma$ by using the verification algorithm of our one-message NIWI construction in Section E-D for the relation $\mathsf{R}^{\mathsf{dec}}$ over bilinear groups in Section E-F. Output the verification result.
  4) If $y = \perp$, then, if for all $j \in [N], \mathsf{Blt}_j = \perp$, output $\mathsf{OK}$, else output $\perp$.

Fig. 8. Our efficient instantiation of eVote $\mathsf{EVOTE}$

settings. Let $(p, \mathbb{G}, \mathbb{G}_t, \mathbf{e}, g)$ be a pairing group setup. Let $X_1, \ldots, X_m \in \mathbb{G}$ and $x_1, \ldots, x_{m'} \in \mathbb{Z}_p$ be variables. Groth and Sahai show how to compute proofs for the following types of equations:

- Pairing product equation. A pairing product equation is an equation of the form

$$\prod_{i=1}^{m} \mathbf{e}(X_i, B_i) \prod_{i=1}^{m} \prod_{j=1}^{m} \mathbf{e}(X_i, X_j)^{\lambda_{ij}} = \mathbf{e}(R, S)$$

  with constants $B_i, R, S \in \mathbb{G}$ and $\lambda_{ij} \in \mathbb{Z}_p$.
- Multi-scalar multiplication equation. A multi-scalar multiplication equation is an equation of the form

$$\prod_{i=1}^{m'} B_i^{x_i} \prod_{i=1}^{m} X_i^{b_i} \prod_{i=1}^{m} \prod_{j=1}^{m'} X_i^{x_i \lambda_{ij}} = T$$

  with constants $B_i, T \in \mathbb{G}$ and $b_i, \lambda_{ij} \in \mathbb{Z}_p$.
- Quadratic equation. A quadratic equation is an equation of the form

$$\sum_{i=1}^{m'} x_i b_i + \sum_{i=1}^{m'} \sum_{i=1}^{m'} \lambda_{ij} x_i x_j \equiv t \bmod p$$

  with constants $b_i, \lambda_{ij}, t \in \mathbb{Z}_p$.

We remark that, in our instantiation, we will only consider pairing product equations and thus we do not need to take into account the issues pointed out by Ghadafi *et al.* [43].

It is possible to compute a proof of the disjunction of two or more pairing product equations [82]. Consider a simple example of two pairing product equations $\mathbf{e}(X_0, B_0) = \mathbf{e}(R_0, S_0)$ and $\mathbf{e}(X_1, B_1) = \mathbf{e}(R_1, S_1)$ where $X_0, X_1 \in \mathbb{G}$ are variables and $B_0, R_0, S_0, B_1, R_1, S_1 \in \mathbb{G}$ are constants. We wish to prove satisfiability of either the first or the second equation.

(The reader may notice that when the constants are different from 1, any instance $B_0, R_0, S_0, B_1, R_1, S_1 \in G$ belongs to the language defined by the relation. Therefore, the trivial non-interactive system that outputs an empty proof and verifies it in the obvious way would be a valid NIWI proof system that satisfies completeness, soundness and WI. Nevertheless, this example is instructive to understand disjunctive proofs and can be generalized to any kind of equation.)

To this end, we add two new variables $\Delta_0$ and $\Delta_1$ and an equation $\mathbf{e}(\Delta_0, g) \cdot \mathbf{e}(\Delta_1, g) = \mathbf{e}(g, g)$. This equation guarantees that at least one of $(\Delta_0, \Delta_1)$ does not equal 1. If $\Delta_0 \neq 1$, we prove satisfiability of the equation $\mathbf{e}(X_0, B_0) = \mathbf{e}(R_0, S_0)$. If $\Delta_1 \neq 1$, we prove satisfiability of the equation $\mathbf{e}(X_1, B_1) = \mathbf{e}(R_1, S_1)$. We also add two variables $\delta_0$ and $\delta_1$ and two equations $\mathbf{e}(\Delta_0, \delta_0) \cdot \mathbf{e}(\Delta_0^{-1}, R_0) = 1$ and $\mathbf{e}(\Delta_1, \delta_1) \cdot \mathbf{e}(\Delta_1^{-1}, R_1) = 1$. The first equation guarantees that, if $\Delta_0 \neq 1$, then $\delta_0 = R_0$, whereas, if $\Delta_0 = 1$, we can set $\delta_0 = 1$. The second equation guarantees that, if $\Delta_1 \neq 1$, then $\delta_1 = R_1$, whereas, if $\Delta_1 = 1$, we can set $\delta_1 = 1$. Finally, we replace $R_0$ and $R_1$ by $\delta_0$ and $\delta_1$ respectively in the original equations. In summary, the OR proof is a proof for the following relation:

$$\begin{aligned} R = \{(w, x) : \\ & \mathbf{e}(\Delta_0, g) \cdot \mathbf{e}(\Delta_1, g) = \mathbf{e}(g, g) \ \wedge \\ & \mathbf{e}(\Delta_0, \delta_0) \cdot \mathbf{e}(\Delta_0^{-1}, R_0) = 1 \ \wedge \\ & \mathbf{e}(\Delta_1, \delta_1) \cdot \mathbf{e}(\Delta_1^{-1}, R_1) = 1 \ \wedge \\ & \mathbf{e}(X_0, B_0) = \mathbf{e}(\delta_0, S_0) \ \wedge \ \mathbf{e}(X_1, B_1) = \mathbf{e}(\delta_1, S_1)\} \end{aligned}$$

Groth-Sahai proofs are proofs about committed values. In order to compute a proof, first one must compute commitments to the variables $X_1, \ldots, X_m \in \mathbb{G}$ and $x_1, \ldots, x_{m'} \in \mathbb{Z}_p$. To this end, the common reference string includes a commitment key. There are two types of keys: a perfectly binding key, which allows the computation of perfectly binding commitments, and a perfectly hiding key, which allows the computation of perfectly hiding commitments. A common reference string contains a key for one of the types. The witness-indistinguishability property of Groth-Sahai proofs holds under the assumption that a perfectly binding key and a perfectly hiding key are computationally indistinguishable.

In the instantiations of Groth-Sahai proofs based on the DLIN assumption, these commitment keys are computed as follows. Pick random $x, y \leftarrow \mathbb{Z}_p$ and compute $f \leftarrow g^x$ and $h \leftarrow g^y$. Pick random $r, s \in \mathbb{Z}_p$ and compute $u \leftarrow f^r$, $h \leftarrow h^s$ and $w \leftarrow g^{r+s}$. A perfectly binding key is $(g, f, h, u, v, w)$, while a perfectly hiding key is $(g, f, h, u, v, wg^{-1})$. Thus, in the Groth-Sahai proof system, we define a common reference string generator $\mathsf{NIWIKeygen}_b$ for the binding setting and a common reference string generator $\mathsf{NIWIKeygen}_h$ for the hiding setting. Moreover, in the binding setting, the system has *perfect soundness*.

A commitment to a variable $X \in \mathbb{G}$ is computed as follows. Pick random $s_1, s_2, s_3 \leftarrow \mathbb{Z}_p$. A perfectly binding commitment is $C = (f^{s_1} u^{s_3}, h^{s_2} v^{s_3}, X g^{s_1+s_2} w^{s_3})$, which is a DLIN encryption of $X$. A perfectly hiding commitment is $C = (f^{s_1} u^{s_3}, h^{s_2} v^{s_3}, X g^{-s_3} g^{s_1+s_2} w^{s_3})$.

A commitment to a variable $x \in \mathbb{Z}_p$ is computed as follows. Pick random $s_1, s_2 \leftarrow \mathbb{Z}_p$. A perfectly binding commitment is $(u^x f^{s_1}, v^x h^{s_2}, w^x g^x g^{r_1+r_2})$. A perfectly hiding commitment is $(u^x f^{s_1}, v^x h^{s_2}, w^x g^{r_1+r_2})$.

### *D. One-message NIWI for the Satisfiability of Equations Over Bilinear Groups*

We give a construction of a one-message NIWI for satisfiability of equations over bilinear groups. We follow the construction in [83]. In [83], a one-message NIWI for $\mathsf{CircuitSat}$ is constructed by using as building blocks two NIZK proofs for $\mathsf{CircuitSat}$. The NIZK proofs use two correlated common reference strings (CRS) and the verifier can check that at least one of them contains a perfectly binding commitment key. The prover computes one NIZK proof for each of the common reference strings. Perfect soundness is guaranteed by the fact that one of the common reference strings contains a perfectly binding commitment key. The common reference string that

contains a perfectly hiding commitment key allows one to prove witness indistinguishability.

The one-message NIWI for CircuitSat in [83] can be used to construct NIWI proofs for the relations $\mathsf{R}^{\mathsf{enc}}$ and $\mathsf{R}^{\mathsf{dec}}$. However, the resulting NIWI proofs are inefficient.

In [32], Groth and Sahai provide NIWI and NIZK proofs for the satisfiability of equations over bilinear groups. The NIWI proofs in [32] use a CRS as trust assumption, and therefore we cannot use them directly to construct NIWI proofs for the relations $\mathsf{R}^{\mathsf{enc}}$ and $\mathsf{R}^{\mathsf{dec}}$. However, we observe that the NIWI and NIZK proofs in [32] are also computed by using two types of CRS: one type contains a perfectly binding commitment key and the other one a perfectly hiding commitment key. Moreover, as in [83], both CRS are indistinguishable but it is possible to check that at least one of the commitment keys is perfectly binding. Consequently, our idea is to construct a one-message NIWI for the satisfiability of equations over bilinear groups by using as building blocks two Groth-Sahai NIWI proofs, one with a CRS that contains a perfectly binding commitment key and another one with a CRS that contains a perfectly hiding commitment key. We remark that, in [83], the prover computes two NIZK proofs instead of NIWI proofs, but we observe that witness-indistinguishability is sufficient. Ràfols [84] already observed that Groth-Sahai proofs can be boosted to non-interactive zaps.

Let $R$ be a relation for which Groth-Sahai NIWI proofs can be computed. Let $\mathsf{NIWI} = (\mathsf{NIWISetup}, \mathsf{NIWIKeygen}_b, \mathsf{NIWIKeygen}_h, \mathsf{NIWIProve}, \mathsf{NIWIVerify})$ be the algorithms of a Groth-Sahai NIWI proof system for relation $R$. NIWI has verifiable correlated key generation if there exist two efficient algorithms $K$ and $V$ with the following properties. We require perfect correctness, i.e., $V$ always accepts the output of $K$ given that $gk$ and $sk$ are honestly generated. We also require soundness, i.e., that for all strings $gk, sk, crs_0, crs_1$ if $V(crs_0, crs_1) = 1$ then either $crs_0$ is a perfectly binding key associated with public parameter $gk$ or $crs_1$ is a perfectly binding key associated with public parameter $gk$. Furthermore, we require that, given $(gk, sk) \leftarrow \mathsf{NIWISetup}(1^k)$, $K(gk, sk)$ outputs two strings $crs_0$ and $crs_1$ such that $crs_0$ (resp. $crs_1$) has the same distribution of the common reference strings output by $\mathsf{NIWIKeygen}_b(gk, sk)$ (resp. $\mathsf{NIWIKeygen}_h(gk, sk)$).

In Groth *et al.* [83], it is shown how to achieve verifiable correlated key generation. Essentially, Groth *et al.* generate two commitment keys $crs_0$ and $crs_1$ that are the same strings, except for the last elements $w_0, w_1$ for which it has to hold that $w_1 = w_0 \cdot g$. (We note that it can be efficiently verified whether a bilinear setup has been generated correctly and that in our case we do not require that $K$ also outputs a trapdoor.) We observe that the common reference string of Groth-Sahai proofs, when instantiated based on the DLIN assumption, contains a commitment key that is equal to the commitment key in [83], and so the property of verifiable correlated key generation also holds for Groth-Sahai proofs.[9]

We describe now the algorithms $(\mathsf{Prove}^R, \mathsf{Verify}^R)$ defined in Section B of a one-message NIWI proof system (*without* CRS) for satisfiability of equations over bilinear groups. $\mathsf{Prove}^R$ and $\mathsf{Verify}^R$ receive as input $(gk, sk)$ output by $\mathsf{NIWISetup}(1^k)$. In our instantiation of EVOTE in Section E-A, $gk$ is replaced by the pairing group setup $\Gamma$ contained in the public key $\mathsf{Pk} \leftarrow (\Gamma, \mathcal{E}.\mathsf{Pk}_1, \ldots, \mathcal{E}.\mathsf{Pk}_4, Z)$, while $sk$ is $\perp$. Let $(gk, x, w)$ belong to $R$.

- $\mathsf{Prove}^R(gk, sk, x, w)$: On input the setup $(gk, sk)$, the instance $x$ and the witness $w$, do the following:
  1) Compute two common reference strings $(crs_0, crs_1) \leftarrow K(gk, sk)$.
  2) Compute a proof $\pi_0 \leftarrow \mathsf{NIWIProve}(gk, crs_0, x, w)$.
  3) Compute a proof $\pi_1 \leftarrow \mathsf{NIWIProve}(gk, crs_1, x, w)$.
  4) Output the proof $\pi \leftarrow (crs_0, crs_1, \pi_0, \pi_1)$.
- $\mathsf{Verify}^R(gk, sk, x, \pi)$: On input the setup $(gk, sk)$, the instance $x$ and the proof $\pi$, do the following:
  1) Output $\perp$ if $0 = V(gk, sk, crs_0, crs_1)$.
  2) Output $\perp$ if $0 = \mathsf{NIWIVerify}(gk, crs_0, x, \pi_0)$.
  3) Output $\perp$ if $0 = \mathsf{NIWIVerify}(gk, crs_1, x, \pi_1)$.
  4) Else, output OK.

We follow the reasoning in [83] to show that this one-message NIWI fulfills the properties of perfect completeness, perfect soundness and computational witness indistinguishability but we simplify the analysis for the latter property.

**Completeness.** The protocol is perfectly complete because the NIWI proofs for the satisfiability of equations over bilinear groups are perfectly complete both on perfectly binding keys and perfectly hiding keys and the verifiable correlated key generation has perfect correctness.

**Soundness.** Perfect soundness follows from the fact that if $V(gk, sk, crs_0, crs_1) = 1$, then either $crs_0$ or $crs_1$ must be a perfectly binding key. The perfect soundness of the proof system over this CRS then implies that the equations must be satisfiable.

**WI.** We now argue that our one-message NIWI is computationally witness indistinguishable assuming verifiable correlated key generation for the homomorphic proof commitment scheme by means of a hybrid argument. The adversary generates an instance $x$ and two witnesses $w_0$ and $w_1$.

1) Hybrid 1. This corresponds to an experiment in which the keys are generated using $K(gk, sk)$ and the proof is computed with witness $w_0$.
2) Hybrid 2. The second hybrid proceeds as the first, except that $crs_0$ is generated using the simulator guaranteed by the composable witness indistinguishability of NIWI. The computational indistinguishability of hybrid 1 and 2 follows from the composable witness indistinguishability of NIWI.

[9] We note that the perfectly binding commitment key (resp. perfectly hiding commitment key) in [83] is the perfectly hiding commitment key (resp. perfectly binding commitment key) in [32]. The reason is the way the commitments are computed in [83] and in [32]. Nevertheless, the arguments to show that the commitment keys satisfy the property of verifiable correlated key generation given in [83] also hold for [32].

3) Hybrid 3. The third hybrid proceeds as the first, except that $\pi_0$ is generated by using witness $w_1$ instead of using witness $w_0$. Hybrid 2 and Hybrid 3 are identically distributed; this follows from the composable witness indistinguishability of NIWI on simulated common reference string.
4) Hybrid 4. The fourth hybrid proceeds as the third, except that $crs_1$ is generated using the simulator guaranteed by the composable witness indistinguishability of NIWI. The computational indistinguishability of hybrid 3 and 4 follows from the composable witness indistinguishability of NIWI.
(Actually, if we used the fact that $crs_1$ is a key for a perfectly hiding commitment, we could conclude that the two experiments are identically distributed but here we are using the composable witness indistinguishability without taking advantage of the two settings.)
5) Hybrid 5. The fifth hybrid proceeds as the fourth, except that $\pi_1$ is generated by using witness $w_1$ instead of $w_0$. Hybrid 4 and Hybrid 5 are identically distributed; this follows from the composable witness indistinguishability of NIWI on simulated common reference string. The computational indistinguishability of hybrid 3 and 4 follows from the composable witness indistinguishability of NIWI.

The indistinguishability of the above hybrid experiments implies the computational witness indistinguishability of NIWI.

### *E. Efficiency of Groth-Sahai NIWI proofs based on DLIN*

We describe the computation and communication cost of Groth-Sahai NIWI proofs based on DLIN. For a paring group setup $\Gamma = (p, \mathbb{G}, \mathbb{G}_t, \mathbf{e}, g) \leftarrow \mathcal{G}(1^\lambda)$, let $|\mathbb{G}|$, $|\mathbb{G}_t|$ and $|\mathbb{Z}_p|$ denote the bit size of elements of $\mathbb{G}$, $\mathbb{G}_t$ and $\mathbb{Z}_p$ respectively. We denote by $|\mathsf{exp}|$, $|\mathsf{mul}|$ and $|\mathsf{map}|$ the time in seconds needed to compute an exponentiation, a multi-exponentiation and a bilinear map respectively. We omit faster operations such as multiplication.

We focus our analysis on pairing product equations because this is the type of equation used in the relations $\mathsf{R}^{\mathsf{enc}}$ and $\mathsf{R}^{\mathsf{dec}}$ described in Section E-F.

We distinguish two types of pairing product equation: linear and quadratic. For variables $X_1, \ldots, X_m \in \mathbb{G}$, a linear pairing product equation is of the form

$$\prod_{i=1}^{m} \mathbf{e}(X_i, B_i) = \mathbf{e}(R, S)$$

with constants $B_i, R, S \in \mathbb{G}$.

For variables $X_1, \ldots, X_m \in \mathbb{G}$, a quadratic pairing product equation is of the form

$$\prod_{i=1}^{m} \mathbf{e}(X_i, B_i) \prod_{i=1}^{m} \prod_{j=1}^{m} \mathbf{e}(X_i, X_j)^{\lambda_{ij}} = \mathbf{e}(R, S)$$

First, we analyze the size of the proof. The common reference string consists of 6 group elements. The size of two correlated common reference strings, i.e., two common reference strings where one contains a perfectly hiding commitment key and the other one a perfectly binding commitment key and that share all the elements except the last one, is $7 \cdot |\mathbb{G}|$. The size of a proof is independent of whether they are computed on input a perfectly binding or a perfectly hiding commitment key. For a relation that involves $K$ linear pairing product equations, a proof $\gamma = (\bar{d}, \bar{\phi})$ consists of a vector $\bar{d}$ of $m$ commitments to the variables $X_1, \ldots, X_m$ and a vector $\bar{\phi}$ of dimension $K$, where each component consists of three group elements. Therefore, the size of the proof for a relation with $m$ variables and $K$ linear pairing product equations is $3(m+K) \cdot |\mathbb{G}|$. For a relation that involves $K$ quadratic pairing product equations, a proof $\gamma = (\bar{d}, \bar{\phi})$ consists of a vector of $m$ commitments to the variables $X_1, \ldots, X_m$ and a vector $\bar{\phi}$ of dimension $K$, where each component is a matrix of group elements of dimension $3 \times 3$. Therefore, the size of the proof is $(3m + 9K) \cdot |\mathbb{G}|$. If a relation with $m$ variables combines $K_1$ linear equations and $K_2$ quadratic equations, the size of the proof is $(3(m + K_1) + 9K_2) \cdot |\mathbb{G}|$.

We analyze now the computation cost. The computation time of two correlated common reference strings is $5 \cdot |\mathsf{exp}|$. The computation time of a proof is independent of whether it is computed on input a perfectly binding or a perfectly hiding commitment key.

For a linear pairing product equation with variables $X_1, \ldots, X_m$ and constants $B_1, \ldots, B_m$, a proof $\gamma = (\bar{d}, \bar{\phi})$ consists of a vector $\bar{d}$ of $m$ commitments to the variables $X_1, \ldots, X_m$ and a vector $\bar{\phi}$. The computation time of the commitments $\bar{d}$ is $(3m) \cdot |\mathsf{mul}|$. The computation time of $\bar{\phi}$ is $3 \cdot |\mathsf{mul}|$. (We note that the complexity of each of the multi-exponentiations needed to compute $\bar{\phi}$ depends on $m$.) The verification time of a linear pairing product equation is $(3m+9) \cdot |\mathsf{map}|$. If a relation contains $K$ equations and $m$ variables, the computation time of $\bar{\phi}$ and the verification time for each of the equations depends on the number of variables $m' \leq m$ that are actually involved in each of the equations.

For a quadratic pairing product equation, a proof $\gamma = (\bar{d}, \bar{\phi})$ consists of a vector $\bar{d}$ of $m$ commitments to the variables $X_1, \ldots, X_m$ and a vector $\bar{\phi}$. The computation time of the commitments $\bar{d}$ is $(3m) \cdot |\mathsf{mul}|$. The computation time of $\bar{\phi}$ is $(24) \cdot |\mathsf{mul}|$. (We note that the complexity of some of the multi-exponentiations needed to compute $\bar{\phi}$ depends on $m$.) The verification time is $(3m + 9n + 27) \cdot |\mathsf{map}|$, where we denote by $m$ the number of pairings in the equation that take as input one variable and by $n$ the number of pairings in the equation that take as input two variables.

### *F. Groth-Sahai NIWI Proofs for $\mathsf{R}^{\mathsf{enc}}$ and $\mathsf{R}^{\mathsf{dec}}$*

**Groth-Sahai NIWI Proofs for $\mathsf{R}^{\mathsf{enc}}$.** We describe an instantiation of the relation $\mathsf{R}^{\mathsf{enc}}$ as a set of pairing product equations over bilinear groups. Groth-Sahai NIWI proofs for this instantiation exist. For clarity of exposition, we describe first a simple relation, which we augment step by step until showing the instantiation of relation $\mathsf{R}^{\mathsf{enc}}$.

Consider the DLIN public key encryption scheme. Let $\Gamma = (p, \mathbb{G}, \mathbb{G}_t, \mathbf{e}, g)$ be a paring product setup. Let $\mathcal{E}.\mathsf{Pk} \leftarrow$

Witness: $w = (\Delta_2, \Delta_3, \alpha_4, \beta_4, \delta_{30}, \delta_{31}, \delta_{32}, [\alpha_l, \beta_l, m_l, \delta_{20l}, \delta_{21l}, \delta_{22l}]_{l=1}^{3}, \delta_2, \Delta_0, \Delta_1, \delta_{00}, \delta_{01}, \delta_{02}, \delta_{10}, \delta_{11}, \delta_{12})$

Instance: $x = (\Gamma, [g_l, f_l, h_l, a_l, b_l, c_l]_{l=1}^{4})$

$$
\begin{aligned}
\mathsf{R}^{\mathsf{enc}} = \{(w,x) : \\
&\mathbf{e}(\Delta_2, g) \cdot \mathbf{e}(\Delta_3, g) = \mathbf{e}(g,g) \ \wedge \ \mathbf{e}(\Delta_3^{-1}, \delta_{30}) \cdot \mathbf{e}(\Delta_3, a_4) = 1 \ \wedge \\
&\mathbf{e}(\Delta_3^{-1}, \delta_{31}) \cdot \mathbf{e}(\Delta_3, b_4) = 1 \ \wedge \ \mathbf{e}(\Delta_3^{-1}, \delta_{32}) \cdot \mathbf{e}(\Delta_3, c_4) = 1 \ \wedge \\
&\mathbf{e}(f_4, \alpha_4) = \mathbf{e}(\delta_{30}, g_4) \ \wedge \ \mathbf{e}(h_4, \beta_4) = \mathbf{e}(\delta_{31}, g_4) \ \wedge \\
&\mathbf{e}(g, \alpha_4) \cdot \mathbf{e}(g, \beta_4) = \mathbf{e}(g, \delta_{32}) \ \wedge \ \mathbf{e}(\Delta_2^{-1}, \delta_2) \cdot \mathbf{e}(\Delta_2, g) = 1 \ \wedge \\
&\bigwedge_{l=1}^{3} [\mathbf{e}(\Delta_2^{-1}, \delta_{20l}) \cdot \mathbf{e}(\Delta_2, a_l) = 1 \ \wedge \ \mathbf{e}(\Delta_2^{-1}, \delta_{21l}) \cdot \mathbf{e}(\Delta_2, b_l) = 1 \ \wedge \\
&\mathbf{e}(\Delta_2^{-1}, \delta_{22l}) \cdot \mathbf{e}(\Delta_2, c_l) = 1 \ \wedge \\
&\mathbf{e}(f_l, \alpha_l) = \mathbf{e}(\delta_{20l}, g_l) \ \wedge \ \mathbf{e}(h_l, \beta_l) = \mathbf{e}(\delta_{21l}, g_l) \ \wedge \\
&\mathbf{e}(g, \alpha_l) \cdot \mathbf{e}(g, \beta_l) \cdot \mathbf{e}(g, m_l) = \mathbf{e}(g, \delta_{22l})] \ \wedge \\
&\mathbf{e}(g, m_1) \cdot \mathbf{e}(g^{-1}, m_2) = 1 \ \wedge \ \mathbf{e}(g, m_1) \cdot \mathbf{e}(g^{-1}, m_3) = 1 \ \wedge \\
&\mathbf{e}(\Delta_0, g) \cdot \mathbf{e}(\Delta_1, g) = \mathbf{e}(\delta_2, g) \ \wedge \\
&\mathbf{e}(\Delta_0^{-1}, \delta_{00}) \cdot \mathbf{e}(\Delta_0, c_1) = 1 \ \wedge \ \mathbf{e}(\Delta_0^{-1}, \delta_{01}) \cdot \mathbf{e}(\Delta_0, \alpha_1) = 1 \ \wedge \\
&\mathbf{e}(\Delta_0^{-1}, \delta_{02}) \cdot \mathbf{e}(\Delta_0, \beta_1) = 1 \ \wedge \ \mathbf{e}(g, \delta_{01}) \cdot \mathbf{e}(g, \delta_{02}) = \mathbf{e}(\delta_{00}, g) \ \wedge \\
&\mathbf{e}(\Delta_1^{-1}, \delta_{10}) \cdot \mathbf{e}(\Delta_1, g^{-1} c_1) = 1 \ \wedge \ \mathbf{e}(\Delta_1^{-1}, \delta_{11}) \cdot \mathbf{e}(\Delta_1, \alpha_1) = 1 \ \wedge \\
&\mathbf{e}(\Delta_1^{-1}, \delta_{12}) \cdot \mathbf{e}(\Delta_1, \beta_1) = 1 \ \wedge \ \mathbf{e}(g, \delta_{11}) \cdot \mathbf{e}(g, \delta_{12}) = \mathbf{e}(\delta_{10}, g)\}
\end{aligned}
$$

Fig. 9. Relation $\mathsf{R}^{\mathsf{enc}}$

$$
\begin{aligned}
R_1 = \{(w,x) : \\
&\mathbf{e}(f, \alpha) = \mathbf{e}(a, g) \ \wedge \ \mathbf{e}(h, \beta) = \mathbf{e}(b, g) \ \wedge \\
&\mathbf{e}(g, \alpha) \cdot \mathbf{e}(g, \beta) \cdot \mathbf{e}(g, m) = \mathbf{e}(g, c)\}
\end{aligned}
$$

Fig. 10. Relation $R_1$.

$$
\begin{aligned}
R_2 = \{(w,x) : \\
&\bigwedge_{l=1}^{3} [\mathbf{e}(f_l, \alpha_l) = \mathbf{e}(a_l, g_l) \ \wedge \\
&\mathbf{e}(h_l, \beta_l) = \mathbf{e}(b_l, g_l) \ \wedge \\
&\mathbf{e}(g, \alpha_l) \cdot \mathbf{e}(g, \beta_l) \cdot \mathbf{e}(g, m_l) = \mathbf{e}(g, c_l)] \ \wedge \\
&\mathbf{e}(g, m_1) \cdot \mathbf{e}(g^{-1}, m_2) = 1 \ \wedge \\
&\mathbf{e}(g, m_1) \cdot \mathbf{e}(g^{-1}, m_3) = 1\}
\end{aligned}
$$

Fig. 11. Relation $R_2$.

$(\Gamma, g, f, h)$ be the public key and $\mathcal{E}.\mathsf{Sk} \leftarrow (x, y)$ be the secret key such that $f = g^{1/x}$ and $h = g^{1/y}$. Let $C = (a, b, c) = (f^r, h^s, g^{r+s} \cdot m)$ be a DLIN ciphertext, which can be decrypted by computing $m \leftarrow c/(a^x b^y)$.

Consider the following relation $R_1$ in Fig. 10. The witness is $w = (\alpha, \beta, m)$ for $\alpha = g^r$ and $\beta = g^s$, and the instance is $x = (\Gamma, g, f, h, a, b, c)$. This relation is fulfilled by the message $m$ encrypted in the ciphertext $(a, b, c)$.

In $\mathsf{R}^{\mathsf{enc}}$, we need to prove that three ciphertexts $(a_1, b_1, c_1)$, $(a_2, b_2, c_2)$ and $(a_3, b_3, c_3)$ encrypt the same message. For $l = 1$ to 3, each of the ciphertexts $(a_l, b_l, c_l)$ is computed on input a different public key $\mathcal{E}.\mathsf{Pk}_l = (\Gamma, g_l, f_l, h_l)$.[10] Consider the following relation $R_2$ in Fig. 11. The witness is $w = [\alpha_l, \beta_l, m_l]_{l=1}^{3}$ and the instance is $x = (\Gamma, [g_l, f_l, h_l, a_l, b_l, c_l]_{l=1}^{3})$. This relation is satisfied if $m_1 = m_2 = m_3$.

In $\mathsf{R}^{\mathsf{enc}}$, we need to prove that the encrypted message is in the

[10]For the sake of clarity, we include in each public key a different element $g_l$ but note that, according to the description of the DLIN encryption scheme presented in Section B, all $g_l$'s values correspond to the same group element $g$ that is contained in $\Gamma$.

$$
\begin{aligned}
R_3 = \{(w, x) : \\
&[\mathbf{e}(f_l, \alpha_l) = \mathbf{e}(a_l, g_l) \ \wedge \ \mathbf{e}(h_l, \beta_l) = \mathbf{e}(b_l, g_l) \ \wedge \\
&\mathbf{e}(g, \alpha_l) \cdot \mathbf{e}(g, \beta_l) \cdot \mathbf{e}(g, m_l) = \mathbf{e}(g, c_l)]_{l=1}^{3} \ \wedge \\
&\mathbf{e}(g, m_1) \cdot \mathbf{e}(g^{-1}, m_2) = 1 \ \wedge \\
&\mathbf{e}(g, m_1) \cdot \mathbf{e}(g^{-1}, m_3) = 1 \ \wedge \\
&\mathbf{e}(\Delta_0, g) \cdot \mathbf{e}(\Delta_1, g) = \mathbf{e}(g, g) \ \wedge \\
&\mathbf{e}(\Delta_0^{-1}, \delta_{00}) \cdot \mathbf{e}(\Delta_0, c_1) = 1 \ \wedge \\
&\mathbf{e}(\Delta_0^{-1}, \delta_{01}) \cdot \mathbf{e}(\Delta_0, \alpha_1) = 1 \ \wedge \\
&\mathbf{e}(\Delta_0^{-1}, \delta_{02}) \cdot \mathbf{e}(\Delta_0, \beta_1) = 1 \ \wedge \\
&\mathbf{e}(g, \delta_{01}) \cdot \mathbf{e}(g, \delta_{02}) = \mathbf{e}(\delta_{00}, g) \ \wedge \\
&\mathbf{e}(\Delta_1^{-1}, \delta_{10}) \cdot \mathbf{e}(\Delta_1, g^{-1} c_1) = 1 \ \wedge \\
&\mathbf{e}(\Delta_1^{-1}, \delta_{11}) \cdot \mathbf{e}(\Delta_1, \alpha_1) = 1 \ \wedge \\
&\mathbf{e}(\Delta_1^{-1}, \delta_{12}) \cdot \mathbf{e}(\Delta_1, \beta_1) = 1 \ \wedge \\
&\mathbf{e}(g, \delta_{11}) \cdot \mathbf{e}(g, \delta_{12}) = \mathbf{e}(\delta_{10}, g)\}
\end{aligned}
$$

Fig. 12. Relation $R_3$.

message space $\mathcal{M}$. We consider here the message space $\{0, 1\}$, which we represent as $\{1, g\} \in \mathbb{G}$. Consider the following relation $R_3$ in Fig. 12. The witness is $w = [\alpha_l, \beta_l, m_l]_{l=1}^{3}, \Delta_0, \Delta_1, \delta_{00}, \delta_{01}, \delta_{02}, \delta_{10}, \delta_{11}, \delta_{12})$ and the instance is $x = (\Gamma, [g_l, f_l, h_l, a_l, b_l, c_l]_{l=1}^{3})$. We introduce two variables $\Delta_0$ and $\Delta_1$. These variables are used to compute an OR proof of pairing product equations as described in Section E-C. When $\Delta_0 \neq 1$, relation $R_3$ is satisfied if $m_1 = 1$, whereas if $\Delta_1 \neq 1$, relation $R_3$ is satisfied if $m_1 = g$. If $\Delta_0 \neq 1$, the variables $(\delta_{00}, \delta_{01}, \delta_{02})$ must equal $(c_1, \alpha_1, \beta_1)$, else we can set $(\delta_{00}, \delta_{01}, \delta_{02}) = (1, 1, 1)$. Similarly, if $\Delta_1 \neq 1$, the variables $(\delta_{10}, \delta_{11}, \delta_{12})$ must equal $(g^{-1}c_1, \alpha_1, \beta_1)$, else we can set $(\delta_{10}, \delta_{11}, \delta_{12}) = (1, 1, 1)$.

We show in Figure 9 our instantiation of relation $\mathsf{R}^{\mathsf{enc}}$ over bilinear groups. $\mathsf{R}^{\mathsf{enc}}$ involves a disjunction of two relations. One of them is $R_3$. The other relation involves showing that the commitment $Z$ in the public key is a commitment to 0, which we represent as $1 \in \mathbb{G}$. As described in Section E-A, the public key $\mathsf{Pk} \leftarrow (\Gamma, \mathcal{E}.\mathsf{Pk}_1, \ldots, \mathcal{E}.\mathsf{Pk}_4, Z)$ contains a commitment Z to the bit 1 (not to be confused with the identity of the group), which we represent as $g \in \mathbb{G}$. (We recall that this part of the disjunction is the "trapdoor mode".) $Z$ is a DLIN ciphertext $(a_4, b_4, c_4)$ computed on input a DLIN public key $\mathcal{E}.\mathsf{Pk}_4 = (\Gamma, g_4, f_4, h_4)$.

We introduce the variables $\Delta_2$ and $\Delta_3$. If $\Delta_2 \neq 1$, we are in the real mode, where we prove relation $R_3$, i.e., that the three ciphertexts encrypt the same message and that the message is in $1, g \in \mathbb{G}$. When $\Delta_2 \neq 1$, the variables $[\delta_{20l}, \delta_{21l}, \delta_{22l}]_{l=1}^{3}$ must equal $[a_l, b_l, c_l]_{l=1}^{3}$ and the variable $\delta_2$ must equal $g$. If $\Delta_3 \neq 1$, we are in the trapdoor mode, where we prove that the commitment $(a_4, b_4, c_4)$ is a commitment to $1 \in \mathbb{G}$. When $\Delta_3 \neq 1$, the variables $(\delta_{30}, \delta_{31}, \delta_{32})$ must equal $(a_4, b_4, c_4)$.

**Groth-Sahai NIWI Proofs for $\mathsf{R}^{\mathsf{dec}}$.** We describe an instantiation of the relation $\mathsf{R}^{\mathsf{dec}}$ as a set of pairing product equations over bilinear groups. Groth-Sahai NIWI proofs for this instantiation exist. For clarity of exposition, we describe first a simple relation, which we augment step by step until showing the instantiation of relation $\mathsf{R}^{\mathsf{dec}}$.

Consider the DLIN public key encryption scheme. Let $\Gamma = (p, \mathbb{G}, \mathbb{G}_t, \mathbf{e}, g)$ be a paring product setup. Let $\mathcal{E}.\mathsf{Pk} \leftarrow (\Gamma, g, f, h)$ be the public key and $\mathcal{E}.\mathsf{Sk} \leftarrow (x, y)$ be the secret key such that $f = g^{1/x}$ and $h = g^{1/y}$. Let $C = (a, b, c) = (f^r, h^s, g^{r+s} \cdot m)$ be a DLIN ciphertext, which can be decrypted by computing $m \leftarrow c/(a^x b^y)$.

Consider the following relation $R_1$ in Fig. 14. The witness is $w = (\alpha, \beta, m)$ for $\alpha = a^x$ and $\beta = b^y$, and the instance is $x = (\Gamma, g, f, h, a, b, c)$. This relation is fulfilled by the message $m$ obtained by decrypting the ciphertext $(a, b, c)$.

In $\mathsf{R}^{\mathsf{dec}}$, we need to prove that the encrypted message is in the message space $\mathcal{M}$. We consider here the message space $\{0, 1\}$, which we represent as $\{1, g\} \in \mathbb{G}$. Consider the following relation $R_2$ in Fig. 15. The witness is $w = (\alpha, \beta, m, \Delta_0, \Delta_1, \delta_{00}, \delta_{01}, \delta_{02}, \delta_{10}, \delta_{11}, \delta_{12})$ and the instance is $x = (\Gamma, g, f, h, a, b, c)$. We introduce two variables $\Delta_0$ and $\Delta_1$. When $\Delta_0 \neq 1$, relation $R_2$ is satisfied if $m = 1$, while if $\Delta_1 \neq 1$, relation $R_2$ is satisfied if $m = g$. If $\Delta_0 \neq 1$, the variables $(\delta_{00}, \delta_{01}, \delta_{02})$ must equal $(c_1, \alpha_1, \beta_1)$, else we can set $(\delta_{00}, \delta_{01}, \delta_{02}) = (1, 1, 1)$. Similarly, if $\Delta_1 \neq 1$, the variables $(\delta_{10}, \delta_{11}, \delta_{12})$ must equal $(g^{-1}c_1, \alpha_1, \beta_1)$, else we can set $(\delta_{10}, \delta_{11}, \delta_{12}) = (1, 1, 1)$.

In $\mathsf{R}^{\mathsf{dec}}$, the authority needs to prove that, given three ciphertexts $(a_1, b_1, c_1)$, $(a_2, b_2, c_2)$ and $(a_3, b_3, c_3)$, it decrypts two of them without revealing which two ciphertexts are decrypted. For $l = 1$ to 3, each of the ciphertexts $(a_l, b_l, c_l)$ is computed on input a different public key $\mathcal{E}.\mathsf{Pk}_l = (\Gamma, g_l, f_l, h_l)$. To compute this proof, we use an OR relation. The OR relation shows that either the first and the second ciphertexts are decrypted (case $x$), or that the first and the third ciphertexts are decrypted (case $y$), or that the second and the third ciphertexts are decrypted (case $z$). Consider the following relation $R_3$ in Fig. 16. Let $L_x = \{1, 2\}$, $L_y = \{1, 3\}$ and $L_z = \{2, 3\}$. The witness is $w = (\Delta_x, \Delta_y, \Delta_z, [\Delta_{t,l,0}, \Delta_{t,l,1}, \delta_{t,l,1}, \delta_{t,l,2}, \delta_{t,l,3}, \delta_{t,l,4}, \alpha_{t,l}, \beta_{t,l}, m_{t,l}, \delta_{t,l,00}, \delta_{t,l,01}, \delta_{t,l,02}, \delta_{t,l,10}, \delta_{t,l,11}, \delta_{t,l,12}]_{t \in \{x,y,z\}, l \in L_t})$ and the instance is $x = (\Gamma, [g_l, f_l, h_l, a_l, b_l, c_l]_{l=1}^{3})$. We introduce three variables $\Delta_x$, $\Delta_y$ and $\Delta_z$. If $\Delta_t \neq 1$ for $t \in \{x, y, z\}$, we are in case $t$. When $\Delta_t \neq 1$ for $t \in \{x, y, z\}$, the variables $[\delta_{t,l,1}, \delta_{t,l,2}, \delta_{t,l,3}]_{l \in L_t}$ must equal $(a_l, b_l, c_l)_{l \in L_t}$ and $\delta_{t,l,4}$ must equal $g$. Additionally, when $\Delta_t \neq 1$ for $t \in \{x, y, z\}$, because then $\delta_{t,l,4}$ must equal $g$, at least one of the variables $\Delta_{t,l,0}$ or $\Delta_{t,l,1}$ does not equal 1, so the proof that the encrypted message is in $1, g \in \mathbb{G}$ is computed as shown in relation $R_2$.

In $\mathsf{R}^{\mathsf{dec}}$, the relation $R_3$ must be proven $N$ times, one for each of the ballots. Additionally, the authority must prove that the tally $y$ is computed following the tally function $F$. We use as $F$ the sum of the encrypted votes in $\{0, 1\}$. Because we represent $\{0, 1\}$ as $1, g \in \mathbb{G}$, the tally function is $F = \prod_{i=j}^{N} m_j$, where

Witness: $w = (\Delta_x, \Delta_y, \Delta_z, \delta_x, \delta_y, \delta_z, [\Delta^j_{t,l,0}, \Delta^j_{t,l,1}, \delta^j_{t,l,1}, \delta^j_{t,l,2}, \delta^j_{t,l,3}, \delta^j_{t,l,4}, \alpha^j_{t,l}, \beta^j_{t,l}, m^j_{t,l}, \delta^j_{t,l,00}, \delta^j_{t,l,01}, \delta^j_{t,l,02}, \delta^j_{t,l,10}, \delta^j_{t,l,11}, \delta^j_{t,l,12}]_{t\in\{x,y,z\}, l\in L_t, j\in[N]|\mathsf{Blt}_j\neq\perp})$

Instance: $x = (\Gamma, [g_l, f_l, h_l]_{l\in[3]}, [a^j_l, b^j_l, c^j_l]_{l\in[3], j\in[1,N]|\mathsf{Blt}_j\neq\perp}, y)$

$$
\begin{aligned}
\mathsf{R}^{\mathsf{dec}} = \{(w,x) :\ & \\
& \mathbf{e}(\Delta_x, g)\cdot\mathbf{e}(\Delta_y, g)\cdot\mathbf{e}(\Delta_z, g) = \mathbf{e}(g,g) \ \wedge \\
& \bigwedge_{t\in\{x,y,z\}} [ \\
& \mathbf{e}(\Delta_t, \delta_t)\cdot\mathbf{e}(\Delta_t^{-1}, y) = 1 \ \wedge \\
& \bigwedge_{l\in L_t} [ \bigwedge_{j\in[N]|\mathsf{Blt}_j\neq\perp} [ \\
& \mathbf{e}(\Delta_t, \delta^j_{t,l,1})\cdot\mathbf{e}(\Delta_t^{-1}, a^j_l) = 1 \ \wedge\ \mathbf{e}(\Delta_t, \delta^j_{t,l,2})\cdot\mathbf{e}(\Delta_t^{-1}, b^j_l) = 1 \ \wedge \\
& \mathbf{e}(\Delta_t, \delta^j_{t,l,3})\cdot\mathbf{e}(\Delta_t^{-1}, c^j_l) = 1 \ \wedge\ \mathbf{e}(\Delta_t, \delta^j_{t,l,4})\cdot\mathbf{e}(\Delta_t^{-1}, g) = 1 \ \wedge \\
& \mathbf{e}(\alpha^j_{t,l}, g_l) = \mathbf{e}(\delta^j_{t,l,1}, f_l) \ \wedge\ \mathbf{e}(\beta^j_{t,l}, g_l) = \mathbf{e}(\delta^j_{t,l,2}, h_l) \ \wedge\ \mathbf{e}(g, m^j_{t,l})\cdot\mathbf{e}(g, \alpha^j_{t,l})\cdot\mathbf{e}(g, \beta^j_{t,l}) = \mathbf{e}(g, \delta^j_{t,l,3}) \ \wedge \\
& \mathbf{e}(\Delta^j_{t,l,0}, g)\cdot\mathbf{e}(\Delta^j_{t,l,1}, g) = \mathbf{e}(\delta^j_{t,l,4}, g) \ \wedge\ \mathbf{e}((\Delta^j_{t,l,0})^{-1}, \delta^j_{t,l,00})\cdot\mathbf{e}(\Delta^j_{t,l,0}, c_l) = 1 \ \wedge \\
& \mathbf{e}((\Delta^j_{t,l,0})^{-1}, \delta^j_{t,l,01})\cdot\mathbf{e}(\Delta^j_{t,l,0}, \alpha^j_{t,l}) = 1 \ \wedge\ \mathbf{e}((\Delta^j_{t,l,0})^{-1}, \delta^j_{t,l,02})\cdot\mathbf{e}(\Delta^j_{t,l,0}, \beta^j_{t,l}) = 1 \ \wedge \\
& \mathbf{e}(g, \delta^j_{t,l,01})\cdot\mathbf{e}(g, \delta^j_{t,l,02}) = \mathbf{e}(\delta^j_{t,l,00}, g) \ \wedge\ \mathbf{e}((\Delta^j_{t,l,1})^{-1}, \delta^j_{t,l,10})\cdot\mathbf{e}(\Delta^j_{t,l,1}, g^{-1}c_l) = 1 \ \wedge \\
& \mathbf{e}((\Delta^j_{t,l,1})^{-1}, \delta^j_{t,l,11})\cdot\mathbf{e}(\Delta^j_{t,l,1}, \alpha^j_{t,l}) = 1 \ \wedge\ \mathbf{e}((\Delta^j_{t,l,1})^{-1}, \delta^j_{t,l,12})\cdot\mathbf{e}(\Delta^j_{t,l,1}, \beta^j_{t,l}) = 1 \ \wedge \\
& \mathbf{e}(g, \delta^j_{t,l,11})\cdot\mathbf{e}(g, \delta^j_{t,l,12}) = \mathbf{e}(\delta^j_{t,l,10}, g)] \ \wedge \\
& \prod_{i=1}^{N} \mathbf{e}(m^j_{t,l}, g) = \mathbf{e}(\delta_t, g)]]\}
\end{aligned}
$$

Fig. 13. Relation $\mathsf{R}^{\mathsf{dec}}$

$$
\begin{aligned}
R_1 = \{(w,x) :\ & \\
& \mathbf{e}(\alpha, f) = \mathbf{e}(a, g) \ \wedge\ \mathbf{e}(\beta, h) = \mathbf{e}(b, g) \ \wedge \\
& \mathbf{e}(g, m)\cdot\mathbf{e}(g, \alpha)\cdot\mathbf{e}(g, \beta) = \mathbf{e}(g, c)\}
\end{aligned}
$$

Fig. 14. Relation $R_1$.

$$
\begin{aligned}
R_2 = \{(w,x) :\ & \\
& \mathbf{e}(\alpha, g) = \mathbf{e}(a, f) \ \wedge\ \mathbf{e}(\beta, g) = \mathbf{e}(b, h) \ \wedge \\
& \mathbf{e}(g, m)\cdot\mathbf{e}(g, \alpha)\cdot\mathbf{e}(g, \beta) = \mathbf{e}(g, c) \ \wedge \\
& \mathbf{e}(\Delta_0, g)\cdot\mathbf{e}(\Delta_1, g) = \mathbf{e}(g, g) \ \wedge \\
& \mathbf{e}(\Delta_0^{-1}, \delta_{00})\cdot\mathbf{e}(\Delta_0, c) = 1 \ \wedge \\
& \mathbf{e}(\Delta_0^{-1}, \delta_{01})\cdot\mathbf{e}(\Delta_0, \alpha) = 1 \ \wedge \\
& \mathbf{e}(\Delta_0^{-1}, \delta_{02})\cdot\mathbf{e}(\Delta_0, \beta) = 1 \ \wedge \\
& \mathbf{e}(g, \delta_{01})\cdot\mathbf{e}(g, \delta_{02}) = \mathbf{e}(\delta_{00}, g) \ \wedge \\
& \mathbf{e}(\Delta_1^{-1}, \delta_{10})\cdot\mathbf{e}(\Delta_1, g^{-1}c) = 1 \ \wedge \\
& \mathbf{e}(\Delta_1^{-1}, \delta_{11})\cdot\mathbf{e}(\Delta_1, \alpha) = 1 \ \wedge \\
& \mathbf{e}(\Delta_1^{-1}, \delta_{12})\cdot\mathbf{e}(\Delta_1, \beta) = 1 \ \wedge \\
& \mathbf{e}(g, \delta_{11})\cdot\mathbf{e}(g, \delta_{12}) = \mathbf{e}(\delta_{10}, g)\}
\end{aligned}
$$

Fig. 15. Relation $R_2$.

$m_j$ is a vote. We show in Figure 13 our instantiation of relation $\mathsf{R}^{\mathsf{dec}}$ over bilinear groups. We introduce the variables $\delta_t$ for $t \in \{x, y, z\}$, which equal the tally $y$ if $\Delta_t \neq 1$.

### G. *Efficiency of Our Instantiation of* EVOTE

In this section, we analyze the communication and computational cost of our instantiation of EVOTE described in Section E-A. For a paring group setup $\Gamma = (p, \mathbb{G}, \mathbb{G}_t, \mathbf{e}, g) \leftarrow \mathcal{G}(1^\lambda)$, let $|\mathbb{G}|$, $|\mathbb{G}_t|$ and $|\mathbb{Z}_p|$ denote the bit size of elements of $\mathbb{G}$, $\mathbb{G}_t$ and $\mathbb{Z}_p$ respectively. We denote by $|\mathsf{exp}|$, $|\mathsf{mul}|$ and $|\mathsf{map}|$ the time in seconds needed to compute an exponentiation,

$$
\begin{aligned}
R_3 = \{(w,x) :\ & \mathbf{e}(\Delta_x, g)\cdot\mathbf{e}(\Delta_y, g)\cdot\mathbf{e}(\Delta_z, g) = \mathbf{e}(g,g)\ \wedge \\
& \bigwedge_{t\in\{x,y,z\}}\ \bigwedge_{l\in L_t}[ \\
& \mathbf{e}(\Delta_t, \delta_{t,l,1})\cdot\mathbf{e}(\Delta_t^{-1}, a_l) = 1\ \wedge \\
& \mathbf{e}(\Delta_t, \delta_{t,l,2})\cdot\mathbf{e}(\Delta_t^{-1}, b_l) = 1\ \wedge \\
& \mathbf{e}(\Delta_t, \delta_{t,l,3})\cdot\mathbf{e}(\Delta_t^{-1}, c_l) = 1\ \wedge \\
& \mathbf{e}(\Delta_t, \delta_{t,l,4})\cdot\mathbf{e}(\Delta_t^{-1}, g) = 1\ \wedge \\
& \mathbf{e}(\alpha_{t,l}, g_l) = \mathbf{e}(\delta_{t,l,1}, f_l)\ \wedge \\
& \mathbf{e}(\beta_{t,l}, g_l) = \mathbf{e}(\delta_{t,l,2}, h_l)\ \wedge \\
& \mathbf{e}(g, m_{t,l})\cdot\mathbf{e}(g, \alpha_{t,l})\cdot\mathbf{e}(g, \beta_{t,l}) = \mathbf{e}(g, \delta_{t,l,3})\ \wedge \\
& \mathbf{e}(\Delta_{t,l,0}, g)\cdot\mathbf{e}(\Delta_{t,l,1}, g) = \mathbf{e}(\delta_{t,l,4}, g)\ \wedge \\
& \mathbf{e}(\Delta_{t,l,0}^{-1}, \delta_{t,l,00})\cdot\mathbf{e}(\Delta_{t,l,0}, c_l) = 1\ \wedge \\
& \mathbf{e}(\Delta_{t,l,0}^{-1}, \delta_{t,l,01})\cdot\mathbf{e}(\Delta_{t,l,0}, \alpha_{t,l}) = 1\ \wedge \\
& \mathbf{e}(\Delta_{t,l,0}^{-1}, \delta_{t,l,02})\cdot\mathbf{e}(\Delta_{t,l,0}, \beta_{t,l}) = 1\ \wedge \\
& \mathbf{e}(g, \delta_{t,l,01})\cdot\mathbf{e}(g, \delta_{t,l,02}) = \mathbf{e}(\delta_{t,l,00}, g)\ \wedge \\
& \mathbf{e}(\Delta_{t,l,1}^{-1}, \delta_{t,l,10})\cdot\mathbf{e}(\Delta_{t,l,1}, g^{-1}c_l) = 1\ \wedge \\
& \mathbf{e}(\Delta_{t,l,1}^{-1}, \delta_{t,l,11})\cdot\mathbf{e}(\Delta_{t,l,1}, \alpha_{t,l}) = 1\ \wedge \\
& \mathbf{e}(\Delta_{t,l,1}^{-1}, \delta_{t,l,12})\cdot\mathbf{e}(\Delta_{t,l,1}, \beta_{t,l}) = 1\ \wedge \\
& \mathbf{e}(g, \delta_{t,l,11})\cdot\mathbf{e}(g, \delta_{t,l,12}) = \mathbf{e}(\delta_{t,l,10}, g)]\}
\end{aligned}
$$

Fig. 16. Relation $R_3$.

a multi-exponentiation and a bilinear map respectively. We omit faster operations such as multiplication. In Table II, we summarize the communication cost, and, in Table III, we summarize the computational cost. We note that the tables show the size of one ballot $\mathsf{Blt}_j$ and the execution time of Cast and VerifyBallot on input one ballot. Additionally, the execution time of EvalTally and VerifyTally does not include the $N$ executions of VerifyBallot required to verify the ballots, where $N$ is the number of ballots that do not equal $\perp$.

**Communication Cost of** EVOTE**.** Algorithm $\mathsf{Setup}(1^\lambda)$ outputs a public key $\mathsf{Pk} \leftarrow (\Gamma, \mathcal{E}.\mathsf{Pk}_1, \ldots, \mathcal{E}.\mathsf{Pk}_4, Z)$ and a secret key $\mathsf{Sk} \leftarrow (\mathcal{E}.\mathsf{Sk}_1, \mathcal{E}.\mathsf{Sk}_2)$. The pairing group setup $\Gamma$ contains 1 group element. Each public key $\mathcal{E}.\mathsf{Pk}_l$ (for $l = 1$ to 4) consists of 2 group elements (note that they share the group element $g$ in $\Gamma$), the commitment $Z$ also consists of 3 group elements, and each secret key $\mathcal{E}.\mathsf{Sk}_l$ (for $l = 1$ to 2) consists of 2 elements of $\mathbb{Z}_p$. The total size of the public key $\mathsf{Pk}$ is $12 \cdot |\mathbb{G}|$ bits. The size of the secret key $\mathsf{Sk}$ is $4 \cdot |\mathbb{Z}_p|$ bits.

Algorithm $\mathsf{Cast}(\mathsf{Pk}, j, v)$ outputs a ballot $\mathsf{Blt}_j = (\mathsf{Ct}_{j,1}, \ldots, \mathsf{Ct}_{j,3}, \pi_j)$. The size of each ciphertext $\mathsf{Ct}_{j,l}$ (for $l = 1$ to 3) is 3 group elements. The proof $\pi_j$ consists of two correlated common references strings and two Groth-Sahai NIWI proofs for the relation $\mathsf{R}^{\mathsf{enc}}$. The size of the two correlated common reference strings is $7 \cdot |\mathbb{G}|$. The witness of $\mathsf{R}^{\mathsf{enc}}$ consists of $m = 34$ elements of $\mathbb{G}$. $\mathsf{R}^{\mathsf{enc}}$ consists of $K_1 = 18$ linear pairing product equations and $K_2 = 19$ quadratic pairing product equations. Therefore, the size of one proof is $327 \cdot |\mathbb{G}|$. The total size of $\pi_j$ is $661 \cdot |\mathbb{G}|$. The size of a ballot is $670 \cdot |\mathbb{G}|$.

TABLE II
COMMUNICATION COST OF EVOTE

| | |
|---|---|
| Public key $\mathsf{Pk}$ | $12 \cdot \lvert\mathbb{G}\rvert$ |
| Secret key $\mathsf{Sk}$ | $4 \cdot \lvert\mathbb{Z}_p\rvert$ |
| Ballot $\mathsf{Blt}_j$ | $670 \cdot \lvert\mathbb{G}\rvert$ |
| Proof $\gamma$ | $(139 + 1836 \cdot N) \cdot \lvert\mathbb{G}\rvert$ |

Algorithm $\mathsf{EvalTally}(\mathsf{Pk}, \mathsf{Sk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N)$ outputs the tally $y$ and the proof $\gamma$. The proof $\gamma$ consists of two correlated common references strings and two Groth-Sahai NIWI proofs for the relation $\mathsf{R}^{\mathsf{dec}}$. The size of the two correlated common reference strings is $7 \cdot |\mathbb{G}|$. The witness of $\mathsf{R}^{\mathsf{dec}}$ consists of $m = 6 + 90 \cdot N$ elements of $\mathbb{G}$, where $N$ is the number of ballots that do not equal $\perp$. $\mathsf{R}^{\mathsf{dec}}$ consists of $K_1 = 7 + 36 \cdot N$ linear pairing product equations and $K_2 = 3 + 60 \cdot N$ quadratic pairing product equations. Therefore, the size of one proof is $(66 + 918 \cdot N) \cdot |\mathbb{G}|$. The total size of $\gamma$ is $(139 + 1836 \cdot N) \cdot |\mathbb{G}|$.

Algorithms VerifyBallot and VerifyTally output one bit.

**Computational Cost of** EVOTE**.** Algorithm $\mathsf{Setup}(1^\lambda)$ computes a pairing group setup, four key pairs of the DLIN encryption scheme and one DLIN ciphertext. Each key pair computation requires 2 exponentiations. The DLIN ciphertext computation requires 3 exponentiations. Therefore, the execution time is $11 \cdot |\mathsf{exp}|$ plus the execution time of $\mathcal{G}(1^\lambda)$.

Algorithm $\mathsf{Cast}(\mathsf{Pk}, j, v)$ outputs a ballot $\mathsf{Blt}_j = (\mathsf{Ct}_{j,1}, \ldots, \mathsf{Ct}_{j,3}, \pi_j)$. The computation time of a ciphertext $\mathsf{Ct}_{j,l}$ (for $l = 1$ to 3) is $3 \cdot |\mathsf{exp}|$. The proof $\pi_j$ consists of two correlated common references strings and two Groth-Sahai NIWI proofs for the relation $\mathsf{R}^{\mathsf{enc}}$. The computation time of two correlated common reference strings is $5 \cdot |\mathsf{exp}|$. The witness of $\mathsf{R}^{\mathsf{enc}}$ consists of $m = 34$ elements of $\mathbb{G}$. Therefore, the computation time of the commitments $\bar{d}$ of each of the Groth-Sahai NIWI proofs is $102 \cdot |\mathsf{mul}|$. $\mathsf{R}^{\mathsf{enc}}$ consists of $K_1 = 18$ linear pairing product equations and $K_2 = 19$ quadratic pairing product equations. The computation time of $\bar{\phi}$ for each linear equation is $3 \cdot |\mathsf{mul}|$ and thus the time for 18 equations is $54 \cdot |\mathsf{mul}|$. The computation time of $\bar{\phi}$ for each quadratic equation is $24 \cdot |\mathsf{mul}|$ and thus the time for 19 equations is $456 \cdot |\mathsf{mul}|$. In total, the computation time of a ballot $\mathsf{Blt}_j$ is $14 \cdot |\mathsf{exp}| + 1224 \cdot |\mathsf{mul}|$.

Algorithm $\mathsf{VerifyBallot}(\mathsf{Pk}, j, \mathsf{Blt})$ verifies the proof $\pi_j$ for relation $\mathsf{R}^{\mathsf{enc}}$ in the ballot. This implies verifying two Groth-Sahai NIWI proofs for relation $\mathsf{R}^{\mathsf{enc}}$. $\mathsf{R}^{\mathsf{enc}}$ consists of $K_1 = 18$ linear pairing product equations and $K_2 = 19$ quadratic pairing product equations. The verification time for a linear equation is $(3m + 9) \cdot |\mathsf{map}|$, where $m$ is the number of variables in the equation. From those 18 linear equations, 11 equations contain 2 variables, 4 equations contain 3 variables, and 3 equations contain 4 variables. Therefore, the total verification time for the linear equations is $300 \cdot |\mathsf{map}|$. The verification time for

TABLE III
COMPUTATIONAL COST OF EVOTE

| | |
|---|---|
| $\mathsf{Setup}(1^\lambda)$ $\mathsf{Pk}$ | $11 \cdot \lvert\mathsf{exp}\rvert$ |
| $\mathsf{Cast}(\mathsf{Pk}, j, v)$ | $14 \cdot \lvert\mathsf{exp}\rvert + 1224 \cdot \lvert\mathsf{mul}\rvert$ |
| $\mathsf{VerifyBallot}(\mathsf{Pk}, j, \mathsf{Blt})$ | $2130 \cdot \lvert\mathsf{map}\rvert$ |
| $\mathsf{EvalTally}(\mathsf{Pk}, \mathsf{Sk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N)$ | $(5 + 4 \cdot N) \cdot \lvert\mathsf{exp}\rvert + (222 + 3636 \cdot N) \cdot \lvert\mathsf{mul}\rvert$ |
| $\mathsf{VerifyTally}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y, \gamma)$ | $(414 + 6264 \cdot N) \cdot \lvert\mathsf{map}\rvert$ |

a quadratic equation is $(3m + 9n + 27) \cdot |\mathsf{map}|$, where $m$ is the number of linear pairings and $n$ the number of quadratic pairings. From the 19 quadratic equations, 15 equations have $m = 1$ and $n = 1$ and 4 equations have $m = 0$ and $n = 4$. Therefore, the verification time for the quadratic equations is $765 \cdot |\mathsf{map}|$. In total, the verification time of the proof $\pi_j$ is $2130 \cdot |\mathsf{map}|$.

Algorithm $\mathsf{EvalTally}(\mathsf{Pk}, \mathsf{Sk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N)$ computes the tally $y$ and a proof $\gamma$ for relation $\mathsf{R}^{\mathsf{dec}}$. The computation of $y$ requires decrypting $2N$ ciphertexts. Each ciphertext involves 2 exponentiations. The computation of $\gamma$ involves computing two correlated common references strings and two Groth-Sahai NIWI proofs for the relation $\mathsf{R}^{\mathsf{dec}}$. The computation time of two correlated common reference strings is $5 \cdot |\mathsf{exp}|$. The witness of $\mathsf{R}^{\mathsf{dec}}$ consists of $m = 6 + 90 \cdot N$ elements of $\mathbb{G}$. Therefore, the computation time of the commitments $\bar{d}$ of each of the Groth-Sahai NIWI proofs is $(18 + 270 \cdot N) \cdot |\mathsf{mul}|$. $\mathsf{R}^{\mathsf{dec}}$ consists of $K_1 = 7 + 36 \cdot N$ linear pairing product equations and $K_2 = 3 + 60 \cdot N$ quadratic pairing product equations. The computation time of $\bar{\phi}$ for each linear equation is $3 \cdot |\mathsf{mul}|$ and thus the time for $7 + 36 \cdot N$ equations is $(21 + 108 \cdot N) \cdot |\mathsf{mul}|$. The computation time of $\bar{\phi}$ for each quadratic equation is $24 \cdot |\mathsf{mul}|$ and thus the time for $3 + 60 \cdot N$ equations is $(72 + 1440 \cdot N) \cdot |\mathsf{mul}|$. In total, the computation time for a proof $\gamma$ is $5 \cdot |\mathsf{exp}| + (111 + 1818 \cdot N) \cdot |\mathsf{mul}|$ and the computation time of $\mathsf{EvalTally}(\mathsf{Pk}, \mathsf{Sk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N)$ is $(5 + 4 \cdot N) \cdot |\mathsf{exp}| + (222 + 3636 \cdot N) \cdot |\mathsf{mul}|$.

Algorithm $\mathsf{VerifyTally}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y, \gamma)$ verifies a proof $\gamma$. This implies verifying two Groth-Sahai NIWI proofs for relation $\mathsf{R}^{\mathsf{dec}}$. $\mathsf{R}^{\mathsf{dec}}$ consists of $K_1 = 7 + 36 \cdot N$ linear pairing product equations and $K_2 = 3 + 60 \cdot N$ quadratic pairing product equations. The verification time for a linear equation is $(3m + 9) \cdot |\mathsf{map}|$, where $m$ is the number of variables in the equation. From those $7 + 36 \cdot N$ linear equations, $12 \cdot N$ equations contain 2 variables, $1 + 18 \cdot N$ equations contain 3 variables, $6 \cdot N$ equations contain 4 variables, and 6 equations contain $N + 1$ variables. Therefore, the total verification time for the linear equations is $(90 + 648 \cdot N) \cdot |\mathsf{map}|$. The verification time for a quadratic equation is $(3m + 9n + 27) \cdot |\mathsf{map}|$, where $m$ is the number of linear pairings and $n$ the number of quadratic pairings. From those $3 + 60 \cdot N$ quadratic equations, $3 + 36 \cdot N$ equations have $m = 1$ and $n = 1$, and $24 \cdot N$ equations have $m = 0$ and $n = 2$. Therefore, the verification time for the quadratic equations is $(117 + 2484 \cdot N) \cdot |\mathsf{map}|$. In total, the verification time of the proof $\gamma$ is $(414 + 6264 \cdot N) \cdot |\mathsf{map}|$.