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

Gina Gallegos-Garcia[1], Vincenzo Iovino[2], Alfredo Rial[2], Peter B. Rønne[2], and Peter Y. A. Ryan[2]

[1] Instituto Politecnico Nacional, Mexico
ggallegosg@ipn.mx
[2] University of Luxembourg
vinciovino@gmail.com, {alfredo.rial, peter.roenne, peter.ryan}@uni.lu

Abstract. In e-voting protocol design, cryptographers must balance usability and 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 guarantees are demanded. Typically, corrupt authorities that try to fake the result of the tally must always be detected.

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

The disadvantage of NIZKs 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 for the generation of a CRS. The latter, though a popular methodology for designing secure protocols, has been shown to be unsound.

In this paper, we address the design of e-voting protocols that provide verifiability without any trust assumptions, where verifiability here is meant without eligibility verification. We show that Non-Interactive Witness-Indistinguishable proofs can be used for this purpose. All our e-voting schemes are private under the Decision Linear assumption, while the verifiability holds unconditionally. We first present a general construction that supports any tally function but with the drawback of representing the computation as a circuit. Then, we show how to efficiently instantiate it for specific types of elections through Groth-Sahai proofs.

To our knowledge, this is the first private e-voting scheme with *perfect* universal verifiability, i.e. one in which the probability of a fake tally not being detected is 0, and with *non-interactive* protocols that does not rely on trust assumptions.

Keywords: e-voting, verifiability, witness indistinguishability, bilinear maps.

Table of Contents

1 Introduction

1.1 Background and Statement of the Problem

The parties participating in a standard e-voting protocol are multiple voters and one authority. First, the authority sets up a public key retaining a corresponding secret key. A voter computes a ballot on input the public key of the authority and her intended vote and sends the ballot to a write-only 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. We will provide a formal definition of verifiability that is stronger than previous ones in some respects.

Privacy protection assumes the existence of a trusted authority in many e-voting systems [Cha81,CGS97,DJ01,RS06,Adi08,CCC+09,RT09,JCJ10]. As for schemes that distribute the trust among several authorities, privacy protection still requires that not all of the authorities are corrupt. Nevertheless, *verifiability* (also called integrity) should be guaranteed even if the authorities are corrupt.

Many e-voting systems make use of Non-Interactive Zero-Knowledge Proofs (NIZK) [BFM88,DMP88,RS92,Gol01,DDO+01] to provide verifiability. NIZK must provide two properties: soundness and zero-knowledge. Soundness prevents a corrupt prover from proving a false statement, i.e., a statement for which no witness exists. Zero-knowledge ensures that the verifier does not learn any information about the witness.

Zero-knowledge 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 [BFM88]. 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 [NY90,DDO+01,CS03b], zero-knowledge 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 the NIZK proofs are computed by the sender of ciphertexts. 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 [BR93]. The RO model assumes the availability of a perfect random function available to all parties. NIZKs that use the RO model are constructed following the Fiat-Shamir heuristic [FS87]. To prove that a NIZK proof constructed following this heuristic is zero-knowledge, 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 zero-knowledge to hold. Consequently, all the parties must trust the implementation of a concrete hash function (e.g., SHA-3 [BDPA11]). 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. Currently, this needed trust on the implementation of hash functions does not exist. In fact, different political entities have developed their own hash functions because they do not trust the hash functions designed by others. For instance, the Russian government discourages the use of SHA-3 and encourages the use of its own hash function [Fed12].

Moreover, even when programmability is not needed, the RO methodology has been shown to be unsound [CGH98]. Further problems are known regarding the programmability of the RO in the context of NIZK [GK03,Kal06,BDSG+13]. The current techniques to avoid the need of programmability resort to the CRS model [DFN06,Lin15,CG15,CPSV16].

This motivates our main question: is it possible to design an e-voting scheme that is verifiable without assuming any trust assumption (like CRS and RO)?

In a survey [Lip05], Lipmaa asks whether Non-Interactive Witness Indistinguishable Proofs (NIWI) can be used to replace NIZKs. NIWIs can be constructed without using any trust assumptions [GOS06,DN00,BOV03,BP15].[3]

NIWI is a non-interactive proof/argument system that provides weaker security guarantees in comparison to NIZKs. While NIZKs ensure that a proof does not reveal any information about the witness, NIWIs only guarantee that, for any two witnesses w_1 and w_2 for the same statement, a proof computed with w_1 is computationally indistinguishable from a proof computed with 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. Nonethe-

[2] We would like to remark that even under the trust assumption of the RO model, it is not known how to obtain efficient NIZK *proofs* in the RO model. In fact, NIZK systems resulting from 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 NIWIs in the CRS model, like the ones of Groth and Sahai [GS08], and one-message NIWIs 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 NIWIs for CircuitSat of Groth *et al.* [GOS06].

less, inspired by a recent result on functional encryption [BSW11,GGH+13] of Badrinarayanan, Goyal, Jain and Sahai [BGJS16], we are surprisingly able to profitably use NIWI to answer our main question affirmatively.

1.2 Our Results

First, we define correctness, privacy and verifiability properties for an eVote. We define two flavors of privacy and verifiability: weak and full. We propose an eVote that is (fully) private and (fully) verifiable and supports any tally function (representable as a polynomial-sized Boolean circuit). Its privacy can be reduced to the Decision Linear assumption [BBS04]. Its verifiability is *perfect* (see below) and thus is not based on any assumption. Moreover, its verifiability is *universal*, i.e., even a third party who did not participate in the election process should be able to verify the correctness of the tally. As a warm-up, we also describe an eVote that fulfills the weak privacy and weak verifiability properties.

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 [BF03,Jou04]. For instance, we can instantiate our construction with the NIWI of Groth, Ostrovsky and Sahai [GOS06] and the Decision Linear encryption scheme of Boneh *et al.* [BBS04]. When instantiated with those building blocks, our construction is the first eVote with *non-interactive* algorithms for casting and verifying ballots and for computing and verifying the tally that provides *perfect* (weak and full) verifiability (as defined in Def. 3) and that fulfills the (weak and full) privacy property under the Decision Linear assumption [BBS04]. The Decision Linear assumption is a well-studied assumption over bilinear groups. Our construction attains *universal* verifiability, i.e., even third parties who did not participate in the election process are able to verify the tally.

We prove that our weakly verifiable eVote fulfills the weak verifiability and weak privacy properties in Corollary 3, and we prove that our (fully) verifiable eVote fulfills the (full) verifiability and (full) privacy properties in Corollary 6. We remark that the computational assumption is only needed to prove that our eVotes fulfill the (weak or full) privacy properties. In contrast, no assumption at all is necessary to prove that they fulfill the (weak or full) verifiability properties.

The latter constructions are aimed at generality and support 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 quite prohibitive in practice. Nevertheless, in Section 6, we provide an efficient instantiation of our (fully) verifiable eVote for the concrete case of the sum function over a binary domain (i.e., referendum). The instantiation makes use of the celebrated Groth-Sahai proofs [GS08,GSW10] that, as for the general approach, may be based only on the Decision Linear assumption.

Our eVote with non-interactive algorithms is the first eVote whose perfect verifiability is not based on any trust and that is provably secure under a well-studied and falsifiable assumption [Nao03]. The latter 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.

In Section 1.7, we outline how to adapt our (fully) verifiable construction 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 from previous approaches to the problem and may be of independent interest. We stress that the multi-string model of Groth and Ostrovsky [GO14], 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 systems. However, we are not concerned about usability and "human-friendly" verifiability, as dealt with in [Riv06,RR06,RRI16]. Furthermore, we only consider traditional e-voting systems and hence we neglect other approaches [KY02,DJ03,Gro04,HRZ10,KSRH12,GIR16].

Our privacy definition is inspired by the one of Benaloh [Ben87], also called "PRIV" in [BCG+15], which we reformulate by using modern terminology and we modify conveniently to withstand the attacks shown in [BCG+15]. We believe that our (fully) verifiable eVote can be proven secure according to other definitions of security, like for instance the one of Chase *et al.* [CKLM13], but we did not investigate the details because it is out of the scope of this initial work.

1.3 Organization

We describe the concept of eVote and its verifiability and privacy properties in Section 1.4. In Section 2, we present detailed definitions of an eVote and of its verifiability and privacy properties. In Section 3 we present the building blocks we will use in our constructions.

In Section 1.5 (resp. Section 1.6) we include all major details needed to understand our construction for a weakly verifiable eVote (resp. (fully) verifiable eVote) and its security properties. In Section 4 (resp. Section 5) we present the full details of our construction for a weakly verifiable eVote (resp. fully verifiable eVote) and of its security properties.

In Section 6 we provide an efficient instantiation, based on Groth-Sahai proofs, of our (fully) verifiable e-voting scheme for the specific case of the sum function over a binary domain (i.e., referendum).

In Section 1.7, we outline how to adapt our (fully) verifiable eVote to a model with multiple authorities and threshold privacy. In this model, the tally evaluation algorithm is run by a set of authorities and privacy must hold if at least one of the authorities is honest.

In Section 1.8 we make some additional remarks about our definitions and in particular about the possibility of re-using the parameters through different elections. In Section 1.9 we discuss relevant related works. Finally, in Section 7

we discuss some future directions in cryptography and e-voting that our work opens up.

1.4 Our Model and Definitions

In this section, we introduce our definitions of privacy and verifiability. We use a simple e-voting model with a single authority. We remark that, even for this model, it was not known how to avoid the use of CRSs or ROs. In Section 1.7, we outline how to adapt our constructions to a model with multiple authorities. Formal definitions of an eVote and of its privacy and verifiability are given in Section 2.1.

We use a general tally function $F : (\mathcal{M} \cup \{\perp\})^N \to \{0,1\}^\star \cup \{\perp\}$, where $\mathcal{M}$ is the message space. The special symbol $\perp$ denotes either an invalid vote or a blank ballot, when it is input to the function, or an error, when it is output by the function. A voter casts $\perp$ to denote a blank ballot, i.e., a valid ballot where no candidate is chosen. A voter abstains from voting by not casting any ballot or by casting an invalid ballot, which is replaced by $\perp$ in the evaluation phase.

Our general tally function F must satisfy a very natural property given in Def. 2. The messages belong to a message space $\mathcal{M}$ that is not specified. As byproduct, our constructions can be instantiated, for instance, to the case of a YES/NO election with the sum as tally function. As shown in [BCG$^+$15], care has to be taken when considering general tally functions.

Privacy Our *privacy* definition is indistinguishability-based and states that no non-uniform 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, 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 $S \subset [N]$. The set S contains the indices of the strings of arbitrary ballots. For each $j \in S$, $m_{0,j} = m_{1,j}$ must hold. For each $j \notin S$, $m_{0,j}, m_{1,j} \in \mathcal{M} \cup \{\perp\}$ must hold. Moreover, we require that for all $d_1, \ldots, d_N \in \mathcal{M} \cup \{\perp\}$, $F(m'_{0,1}, \ldots, m'_{0,N}) = F(m'_{1,1}, \ldots, m'_{1,N})$ must hold, where, for each $j \in S, m'_{0,i} = m'_{1,i} = d_i$ must hold, and for each $j \notin S$, $b \in \{0,1\}, m'_{b,j} = m_{b,j}$ must hold.

Our definition can be viewed as a variant of Benaloh's ballot privacy definition [Ben87] (also called "PRIV" in [BCG+15]) reformulated by using modern terminology and corrected to rule out some known attacks [BCG+15].

We also define *weak privacy*. The difference between the definitions of *weak privacy* and privacy is that, in *weak privacy*, the set S must be empty, i.e., the adversary cannot submit arbitrary ballots.

The privacy definitions that we use here are simple and do not capture vote replay attacks, see e.g. [CS10]. 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.

Verifiability We define a ballot verification and a tally verification algorithm. In our definition of verifiability, 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". More concretely, consider that some ballots are computed by honest voters and are accepted by the ballot verification ballot 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 $y < 3$.

Remarkably, our construction provides *perfect* verifiability. *Perfect* verifiability means that the probability that a malicious authority computes an incorrect tally and a proof that are accepted by the tally verification algorithm is *null*. We stress that in our model we implicitly assume that the public bulletin board (PBB, in short) is trusted (without this, the universal verifiability would degrade to an individual verifiability notion). This seems a minimal assumption and it is not clear whether it would be possible to remove the trust in the PBB.

We also define *weak verifiability*. In *weak verifiability*, the authority can incorrectly claim that $y = \perp$. The second condition described above is still guaranteed for all the tallies $y \neq \perp$. For a weakly verifiable eVote, we only require weak privacy.

Although our weakly verifiable eVote satisfies weaker properties, it represents a worthwhile warm-up. Our (fully) verifiable eVote is based on it, though with some relevant modifications. Our weakly verifiable construction does not need a ballot verification algorithm, but for simplicity we use the same syntax for both the weakly verifiable and (fully) verifiable schemes. In the next subsection, we describe a definition of correctness that is stronger than previous ones. This definition is needed to exclude the case that an eVote that is intuitively not verifiable fulfills formally the definition of verifiability.

In Section 1.7, we outline how to adapt our (fully) verifiable construction to a model with multiple authorities. In this model, the tally is computed by a set of authorities. Privacy must hold if at least one authority is honest. Because this model is not the main focus of our work, we do not present formal definitions or a detailed description of its construction. We note that such a construction would satisfy a different, but still without trust assumptions, definition of verifiability that essentially states that if there is at least one honest voter, the verifiability holds with overwhelming probability over the random coins of such voter, a very minimal assumption.

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. Unconditional eligibility verifiability seems hard or impossible to achieve, since we normally use some commitment, 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.

Our construction can however easily be extended to take into account such attacks by using digital signatures in a standard, but non-perfect, way (see e.g. [CGGI14]). The resulting construction would nonetheless satisfy a meaningful notion of verifiability secure against computationally bounded adversaries not based on any trust assumption, which advances the state of the art. In fact, to our knowledge, it is not even known how to construct an eVote protocol with computational verifiability without trusted parties.

On the Need of a Stronger Correctness Property We justify here why a stronger correctness property is needed. 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.

We explain now that this is an issue. In our weakly verifiable scheme, the ballot verification algorithm accepts any ballot. Therefore, it would be possible to make such a scheme (fully) verifiable by just changing the tally verification algorithm so that it accept $y = \perp$ only when no ballot passes the ballot verification

algorithm. As can be seen, condition (1) in the definition of (full) verifiability (cf. Def. 3) 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 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. 3). 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.* [BCG$^+$15]) this issue has been overlooked. The fully 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.

1.5 Warm-up: Our Weakly Verifiable eVote (Sketch)

In this section, we sketch our construction for a weakly verifiable eVote, i.e. an eVote that fulfills the weak verifiability and weak privacy properties. A (fully) verifiable eVote, which satisfies (full) verifiability and (full) privacy, is presented in Section 1.6. We stress that in practice such weakly verifiable eVote lacks fundamental security guarantees, but nonetheless it serves as a worthwhile warm-up to our (fully) verifiable eVote.

Intuition Our weakly verifiable eVote uses 3 instances of a public key encryption (PKE) scheme in parallel. We require that the PKE scheme fulfills two properties: perfect correctness and unique secret key (see Def. 6). PKE schemes with those properties are known in the literature [DH76,BBS04] and can be constructed, e.g., from the Decision Linear assumption [BBS04]. The voter encrypts her vote 3 times using the PKE scheme *without* adding any proof of ciphertext well-formedness. Therefore, a ballot consists of three ciphertexts.

To compute the tally, the authority proceeds as follows. The authority decrypts the first ciphertext and the second ciphertext in a ballot. The authority replaces decrypted messages that do not belong to the message space by $\perp$. The authority evaluates the tally function twice. First, the authority uses as input the messages encrypted in the first ciphertext of each ballot. Second, it uses the messages encrypted in the second ciphertext of each ballot. If both tallies are equal, the authority outputs the tally along with a proof of correctness, else the authority returns $\perp$ to indicate an error.

The property of unique secret key guarantees that the decrypted message will be unique for each ciphertext. Without this property, it could be possible that a voter cast an invalid ciphertext Ct not in the ciphertext space such that two well-formed secret keys Sk_1 and Sk_2 for the same public key decrypt Ct to different strings. Note that this is not prevented by the correctness, which only takes in account ciphertexts output of the encryption algorithm.

Sketch of the construction Let N be the number of voters and let F be a tally function with message space $\mathcal{M}$. The public key Pk of our eVote consists of the 3 PKs $(\mathsf{Pk}_1, \ldots, \mathsf{Pk}_3)$ of the underlying PKE. The secret key consists of the 3 corresponding SKs $(\mathsf{Sk}_1, \ldots, \mathsf{Sk}_3)$ of the PKE.

Our cast algorithm takes as input the public key $(\mathsf{Pk}_1, \ldots, \mathsf{Pk}_3)$, the index j of the voter[4] (for $j \in [N]$), and a vote v. The cast algorithm outputs a ballot for the j-th voter. Our cast algorithm just encrypts the vote v with the 3 instances of the PKE to produce the ciphertexts $\mathsf{Ct}_1, \ldots, \mathsf{Ct}_3$. The ballot given as output is $\mathsf{Blt} \stackrel{\triangle}{=} (\mathsf{Ct}_1, \ldots, \mathsf{Ct}_3)$.

The tally evaluation algorithm works as follows. For all $j \in [N]$, if the corresponding voter cast her vote, for all $l \in [2]$, decrypt $\mathsf{Ct}_{j,l}$ with 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 j-th voter did not cast her vote, 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 NIWI proof γ of the fact that $x = (\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, \mathsf{Pk}_1, \ldots, \mathsf{Pk}_3)$ satisfies the relation $\mathsf{R}^{\mathsf{dec}}$ in Fig. 1 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 will be essential for privacy.

Note that the proof γ can be computed using as witness the randomness used to compute the public and secret key pairs. Finally, the algorithm outputs the pair (y, γ). The tally verification algorithm verifies (y, γ) by using the verification algorithm of the NIWI system.

Weak Verifiability of the Construction Weak verifiability (cf. Def. 3) requires that, given a public key and a set of messages decrypted from the ballots, the authority cannot output a pair (y, γ), $y \neq \perp$, such that y is an incorrect tally, but γ is accepted by the tally verification algorithm. However, the authority is able to claim that $y = \perp$ even if that is not the correct tally. The construction described above suffers from this problem.

We give a detailed proof that our construction fulfills the weak verifiability property in Theorem 1. In the following, we explain why our construction above fulfills the two conditions required by the weak verifiability property. First, we show that it 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 C_i (possibly computed on input a maliciously generated public key) must be associated with a

[4] 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 $\mathsf{R}^{\mathsf{dec}}(x, w)$:

Instance: $x = (\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, \mathsf{Pk}_1, \ldots, \mathsf{Pk}_3, y)$. (Recall that a ballot is set to $\perp$ if the corresponding voter did not cast her vote.)

Witness: $w = (\mathsf{Sk}'_1, \mathsf{Sk}'_2, s_1, s_2, i_1, i_2)$, where the s_i's are the randomness used to generate the secret key/public key pairs, which is known by the authority that 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 $\mathsf{Pk}_{i_1}, \mathsf{Pk}_{i_2}$ are constructed using honestly generated public and secret key pairs and are equal to $\mathsf{Sk}'_1, \mathsf{Sk}'_2$; 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 $l \in [2]$, Sk'_l decrypts ciphertext Ct_{j,i_l} in 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 Sk'_l decrypts Ct_{j,i_l} to a string $\notin \mathcal{M}$.

Fig. 1. Relation $\mathsf{R}^{\mathsf{dec}}$.

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 use a contradiction to show that our construction fulfills the first condition. Let us assume that there exist two results $y_0, y_1 \neq \perp$ such that $y_0 \neq y_1$, and two proofs γ_0, γ_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.

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 recall that an honest ballot for the j-th voter consists of three ciphertexts that encrypt the same message m. 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, if the claimed tally y does not equal $\perp$, y has to be in the range of the function F restricted to m at index j.

Weak Privacy of the Construction We explain how we prove that our construction fulfills the weak privacy property. The proof consists of a sequence of hybrid experiments [GM84], which are summarized in Table 1.5.

For simplicity, in this sketch we assume that the adversary submits a challenge that consists of two tuples $(m_{0,1}, \ldots, m_{0,N})$ and $(m_{1,1}, \ldots, m_{1,N})$, where

Table 1. Sequence of hybrid games to prove fulfillment of the weak privacy property.

Exp	$(\mathsf{Ct}_{j,1}, \mathsf{Ct}_{j,2}, \mathsf{Ct}_{j,3})$	Sk index	γ	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

each of the messages belongs to the message space $\mathcal{M}$. In the table, the first column shows the name of the hybrid experiment. The second column shows the three messages that are encrypted in the 3 ciphertexts $\mathsf{Ct}_{j,1}, \ldots, \mathsf{Ct}_{j,3}$ contained in the challenge ballot $\mathsf{Blt}_j = (\mathsf{Ct}_{j,1}, \ldots, \mathsf{Ct}_{j,3})$ associated with voter j. The text in blue in the "Sk index" column denotes the indices used as witness in the proof γ. 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 γ. The text in red indicates the difference from the previous hybrid experiment.

The proofs of indistinguishability between the hybrid experiments H_1 and H_2, H_2 and H_3, and H_3 and H_4 are symmetrical to the proofs of indistinguishability between H_4 and H_5, H_5 and H_6, and H_6 and H_7. Therefore, it suffices to explain how we prove indistinguishability between the first four hybrid experiments.

Hybrid H_1 corresponds to the real experiment, except that the challenger sets the bit $b = 0$.

In hybrid H_2, we switch the third message (in red) in any ballot to encrypt $m_{1,j}$. This is possible because the witness used to compute the proof γ does not contain the randomness used to compute the third secret key. Thanks to that, we can show indistinguishability between H_1 and H_2 by using the IND-CPA property of the PKE scheme.

In hybrid H_3, the witness used to compute the proof γ contains the indices $\{1, 3\}$ instead of $\{1, 2\}$. Therefore, γ is in trapdoor mode. The witness-indistinguishability property of the NIWI allows as to show that H_3 cannot be distinguished from H_2. Note that the result of the decryption does not change thanks to the constraint in the weak privacy definition that $F(m_{0,1}, \ldots, m_{0,N}) = F(m_{1,1}, \ldots, m_{1,N})$ must hold.

In hybrid H_4, we switch the second message (in red) in any ballot to encrypt $m_{1,j}$. This is possible because the witness used to compute the proof γ does not contain the randomness used to compute the second secret key. Thanks to that, we can show indistinguishability between H_4 and H_3 by using the IND-CPA property of the PKE scheme.

We remark that, in order to switch the encrypted messages to $m_{1,j}$'s in every ballot, we use a simple property: the witness of the proof γ contains the randomness used to compute *two* of the secret keys. Thanks to that, we can show indistinguishability between H_1 and H_2 and between H_3 and H_4 by using the IND-CPA property of the PKE scheme whose randomness is not needed to compute γ. We point out that, to prove indistinguishability between those hybrid experiments, we need to use N "sub-hybrids". In each "sub-hybrid", we switch the message encrypted in just one ballot.

In Section 4, we present our weakly verifiable eVote in a more detailed manner.

1.6 Our Fully Verifiable eVote (Sketch)

The scheme sketched in Section 1.5 suffers from a severe problem: the authority can claim that the tally is $\perp$ when it is not. That is, there can be two tallies $y_0 \neq \perp$ and $y_1 = \perp$ and two proofs γ_0 and γ_1 such that both proofs are accepted by the tally verification algorithm.

For instance, consider the following case. The ballots submitted by the voters are such that the tally y_1 obtained by evaluating the tally function on input the messages decrypted from the first ciphertext of each ballot equals the tally y_2 obtained when using the second ciphertext of each ballot, but differs from the tally y_3 obtained when using the third ciphertext. Then, by using the indices $(1, 2)$, the authority can prove successfully that the result of the election is $y_1 = y_2$, and by using indices $(1, 3)$, the authority can claim that the result of the election was $\perp$. The voters do not learn the indices that the authority used in the NIWI proof.

This also allows severe DoS attacks. For example, if just one voter submits a wrong ballot that makes the two tallies y_1 and y_2 be different from each other, then an honest authority has to output $\perp$. Furthermore, this scheme only fulfills the weak privacy property, which does not take into account corrupt voters.

Therefore, we propose a scheme that fulfills the (full) verifiability and the (full) privacy properties. This scheme solves the above-mentioned problems in an elegant way.

Next we show a sketch of the scheme for general tally functions and in Section 6 we provide a more efficient instantiatation of it for the concrete case of the sum function over a binary domain (i.e., referendum).

In Section 5, we present our (fully) verifiable eVote in a more detailed manner.

Sketch of the Construction 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 $Z = \mathsf{Com}(1)$.

A ballot consists of three ciphertexts, which are computed as in the weakly verifiable scheme, and of a proof that either the three ciphertexts encrypt the same message in the message space $\mathcal{M} \cup \{\perp\}$ or Z is a commitment to 0. Formally, the ballot contains a NIWI proof for the relation in Fig. 2.

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

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

Witness : $w \stackrel{\triangle}{=} (m, r_1, \ldots, r_3, u)$, where the values r_l's are the random values used to encrypt the ciphertexts Ct_l's and u is the random value used to compute the commitment Z.

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

1. **Real mode.** All 3 ciphertexts $(\mathsf{Ct}_1, \ldots, \mathsf{Ct}_3)$ encrypt the same message in $\mathcal{M} \cup \{\perp\}$.

OR

2. **Trapdoor mode.** Z is a commitment to 0.

Fig. 2. Relation $\mathsf{R}^{\mathsf{enc,full}}$.

The ballot verification algorithm runs the verification algorithm for the NIWI proof system for the relation $\mathsf{R}^{\mathsf{enc,full}}$. (We recall that the ballot verification algorithm of our weakly verifiable eVote accepts any ballot.) The tally evaluation algorithm is the same as in our weakly verifiable eVote.

The tally verification algorithm also follows the one of the weakly verifiable eVote with the following modification. If either (1) not all inputs are $\perp$ and $y = \perp$, or (2) all inputs are $\perp$ and $y \neq \perp$, the tally verification algorithm outputs $\perp$.

We explain the reason for this modification. First, note that, in the (fully) verifiable scheme, the ballots that are rejected by the ballot verification algorithm are replaced by $\perp$ as input to the tally evaluation algorithm. We recall that, in the weakly verifiable scheme, the tally evaluation algorithm is run on input $\perp$ only when voters do not send any ballot.

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). That is, if at least one message is valid, then it has to be "counted".

As we show below, 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 $y = \perp$.

(Full) Verifiability of the Construction We show that our scheme fulfills the (full) verifiability property. This property consists of two conditions described in Section 1.4 and defined in Def. 3.

First, we show that our scheme fulfills the first condition. (A detailed proof is given in Theorem 4.) This condition requires that the authority cannot output two tallies y_1, y_2 such that $y_1 \neq y_2$ and two proofs γ_1 and γ_2 that are accepted by the tally verification algorithm.

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, as in our weakly verifiable eVote, (2) the authority cannot output two tallies y_1, y_2 such that $y_1 \neq y_2, y_1, y_2 \neq \perp$ along with proofs γ_1 and γ_2 that are accepted by the ballot verification algorithm. Therefore, (1) and (2) imply that the authority cannot output two tallies y_1, y_2 such that $y_1 \neq y_2$.

We show now 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. Therefore, the perfect correctness of the NIWI and of the PKE scheme guarantee that an honestly computed ballot[5] for the j-th voter that encrypts message m will always be "counted", i.e., for any (y, γ) pair that is accepted by the tally verification algorithm, y will be compatible with m at index j according to Def. 1.

(Full) Privacy of the Construction We show now that our scheme fulfills the (full) privacy property. Here we summarize the proof. In Section 5.2, 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.

[5] 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.

The next hybrid experiments are similar to the ones used in the security proof of the weakly verifiable scheme. Thanks to the hybrid experiment H^Z, we can still show indistinguishability between those hybrid experiments 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.

There is one difference between the hybrid experiments in the weakly verifiable scheme and in the (fully) verifiable scheme. Namely, in the (fully) verifiable scheme, we have to handle possibly dishonest ballots. In particular, 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 conditioned on the occurrence of $\bar{E}^1$ (i.e., the complement of E^1).

More concretely, the sequence of hybrid experiments after H^Z is as follows. 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.

- 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 γ 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 Sk_1, 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 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.

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 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 $(1, \perp, 1)$.

1.7 eVote with Multiple Authorities and Threshold Privacy

In this section, we sketch how to generalize our (fully) verifiable construction 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 [GO14] 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.

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 (fully) verifiable eVote and, in addition, a commitment com_k to a tuple of N 0's.

A ballot 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 the (fully) verifiable scheme 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 $k \in [m]$, 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 the (fully) verifiable scheme. The proof of correctness of the tally is a proof for the following modified relation: either (the real statement) the witness satisfies the relation $\mathsf{R}^{\mathsf{dec,full}}$ of the (fully) verifiable scheme and com_k is a commitment to 0 OR (the trapdoor statement) 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.

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 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 Blt_j such that 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 the (fully) verifiable eVote in Section 5 and to the fact that, in real mode, the sum of the messages encrypted in a ballot is equal to a number in $\{0, 1\}$.

Privacy of the Construction We use a selectively-secure model [CHK04] 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 5.2, 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 completely diverge from previous approaches to the problem.

1.8 On The Reusability of the Public Parameters

Our definition of verifiability does not prevent the following undesirable case. Consider an ill-formed ballot 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 $y = F(v_1, \ldots, v_N)$ 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 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 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 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 Blt_1, there could exist two ballots Blt_2 and Blt'_2 such that, in an election with ballots Blt_1 and Blt_2, the result is 2, and, in a election with ballots Blt_1 and 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 Pk and all 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 constructions would not be secure. Nevertheless, in our (fully) verifiable construction, 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 [Yao86,LP09] can be based on the minimal assumption of existence of one-way functions, whereas the reusable variant [GKP+13] is known to be implementable only under stronger assumptions. Similarly, in functional encryption, the schemes with bounded security [SS10,GVW12] can be based just on public key encryption, whereas the unbounded secure variants are only known to be implementable under very strong assumptions [GGHZ16]. For instance, the scheme of Sahai and Seyalioglu [SS10] becomes completely insecure when the adversary can decrypt a ciphertext with two different secret keys, exactly as it occurs for our schemes.

1.9 Related Work

Our work is inspired by the work of Badrinarayanan *et al.* [BGJS16], which puts forward the concept of verifiable functional encryption. (We note that the committing IBE of [GH07] 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 [CHK04].

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 (full) 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.* [GGG+14]) 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 [GGH+13], 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 [FLS90]. They can be viewed as a generalization of it.

Kiayias *et al.* [KZZ15] (see also [CZZ+15] for a distributed implementation) put forth a verifiable eVote without trust assumptions that represents a breakthrough along this direction, but diverges from ours in several fundamental aspects:

- It requires interaction between the voters and the board, whereas all our algorithms are *non-interactive*.
- Receipt-freeness, accountability and degree of dependence on secure channels to distribute vote codes are undetermined issues. In our scheme, voters can verify the election if they just know the ballot they cast. In particular, voters do not need to store the randomness used to compute it, and the authority cannot cheat in the tally process.
- It does not achieve *universal verifiability*, whereas ours does.
- Its information-theoretical verifiability is parameterized and depends on the number of honest voters, whereas ours is *perfect*, i.e., the probability of a wrong tally being accepted by the verification algorithm is equal to *zero*.

– Its privacy can be reduced to 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.[6]
– Its definition of verifiability requires an extraction property, whereas ours does not.

Moran and Naor [MN06] 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 [CGK$^+$16] and [BCG$^+$15] for a survey.

We point out that our definition of verifiability is motivated by the guidelines of [CGK$^+$16]. In its formalization, our definition is similar to the ones of [GIR16], the verifiability for multi-input functional encryption of BGJS and the uniqueness of tally of Bernhard *et al.* [BCG$^+$15]. 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.[7] See also [KRS10] for symbolic approaches to verifiability.

Our privacy notion is inspired by the one of Benaloh [Ben87], which is called "PRIV" in [BCG$^+$15]. We reformulate it by using modern terminology and we conveniently modify it to withstand the attacks shown in [BCG$^+$15]. We refer to [BCG$^+$15] for a survey on definitions of privacy for e-voting.

Perfect verifiability and perfect correctness seem incompatible with receipt-freeness [BT94,SK95,MH96,MN06,DKR09,CCFG16]. 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 [CCFG16]. (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.

Recently, Bellare, Fuchsbauer and Scafuro [BFS16] started the study of security of NIZKs in face of public parameter subversion. They showed the impos-

[6] At some point in the security reduction for our (fully) verifiable 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.

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

sibility of attaining subversion soundness while retaining zero-knowledge, thus justifying our need of sidestepping NIZKs.

As we do in our efficient instantiation of Section 6, Ràfols [Ràf15] observed that Groth-Sahai proofs can be boosted to non-interactive zaps.

2 Definitions

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 1 Given a function $F(x_1, \ldots, x_n) : A^n \to B$, 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 $C_{z_1,\ldots,z_n,i_1,\ldots,i_n} \stackrel{\triangle}{=} \{(x_1, \ldots, x_n) | \forall j \in [k], x_{i_j} = z_j\}$.

2.1 E-Voting Schemes

An e-voting scheme (eVote, in short) 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 Σ is the *range of possible results*. The function $F : (\mathcal{M} \cup \{\perp\})^N \to \Sigma \cup \{\perp\}$ 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[8], 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.

[8] 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.

Definition 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 $\mathcal{M} \cup \{\perp\}$, 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 $m_1 = \perp, \ldots, m_N = \perp$.

Before defining formally an eVote, we explain how its algorithms are used to conduct an election.

The voting ceremony. The voting ceremony occurs 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 $\mathsf{VerifyBallot}$. 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 *evaluation tally* algorithm $\mathsf{EvalTally}$ on input the public key, the secret key, and N strings that represent either ballots or $\perp$ symbols appended to the PBB. $\mathsf{EvalTally}$ 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.

Definition 3 [E-voting Scheme] A $(N, \mathcal{M}, \Sigma, F)$-*e-voting scheme* EVOTE for number of voters $N > 1$, domain of valid votes $\mathcal{M}$, range of possible results Σ and tally function $F : (\mathcal{M} \cup \{\perp\})^N \to \Sigma \cup \{\perp\}$ is a tuple

$$\mathsf{EVOTE} \triangleq (\mathsf{Setup}, \mathsf{Cast}, \mathsf{VerifyBallot}, \mathsf{EvalTally}, \mathsf{VerifyTally})$$

of 5 PPT algorithms, where $\mathsf{VerifyBallot}$ and $\mathsf{VerifyTally}$ 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 γ 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 γ of tally correctness, it outputs a value in $\{\mathsf{OK}, \perp\}$.

An eVote must satisfy the following correctness, verifiability, and privacy properties. We also define the weak verifiability and weak privacy properties. A weakly verifiable eVote must satisfy correctness, weak verifiability and weak privacy.

Correctness and verifiability

- *(Perfect) Correctness.* We require the following conditions (1) and (2) to hold.
  1. Let Abst be a special symbol not in $\mathcal{M} \cup \{\perp\}$ that denotes that a voter did not cast her vote.[9] 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 $(y, \gamma) \stackrel{\triangle}{=} \mathsf{EvalTally}(\mathsf{Pk}, \mathsf{Sk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N)$ 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 $S \stackrel{\triangle}{=} \{j |\ \mathsf{Blt}_j \neq \perp \wedge \mathsf{VerifyBallot}(\mathsf{Pk}, j, \mathsf{Blt}_j) = \perp\}$ 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 $(y, \gamma) \stackrel{\triangle}{=} \mathsf{EvalTally}(\mathsf{Pk}, \mathsf{Sk}, \mathsf{Blt}'_1, \ldots, \mathsf{Blt}'_N)$ then $\mathsf{VerifyTally}(\mathsf{Pk}, \mathsf{Blt}'_1, \ldots, \mathsf{Blt}'_N, y, \gamma) = \mathsf{OK}$.
- *Weak verifiability.* We require the following conditions (1) and (2) to hold.

[9] In the following definition, we need Abst to differentiate the case of a voter who did not cast a vote at all (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.

1. For all $\mathsf{Pk} \in \{0,1\}^\star, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N \in \{0,1\}^\star \cup \{\bot\}$, there exist $m_1, \ldots, m_N \in \mathcal{M} \cup \{\bot\}$ such that for all $y \neq \bot$ and γ in $\{0,1\}^\star$, if $S \triangleq \{j |\ \mathsf{Blt}_j \neq \bot \wedge \mathsf{VerifyBallot}(\mathsf{Pk}, j, \mathsf{Blt}_j) = \bot\}$ 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 $\mathsf{Blt}'_j = \bot$ 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 $m_{i_1}, \ldots, m_{i_k} \in \mathcal{M} \cup \{\bot\}$, all $\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N \in \{0,1\}^\star \cup \{\bot\}$ 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 $S \triangleq \{j |\ \mathsf{Blt}_j \neq \bot \wedge \mathsf{VerifyBallot}(\mathsf{Pk}, j, \mathsf{Blt}_j) = \bot\}$ 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 $\mathsf{Blt}'_j = \bot$ otherwise, it holds that: if there exist $y \in \{0,1\}^\star$ 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 $i_1, \ldots, i_k$.

– *Verifiability.* We require the following conditions (1) and (2) to hold.
1. For all $\mathsf{Pk} \in \{0,1\}^\star, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N \in \{0,1\}^\star \cup \{\bot\}$, there exist $m_1, \ldots, m_N \in \mathcal{M} \cup \{\bot\}$ such that for all $y \in \{0,1\}^\star \cup \{\bot\}$ and γ in $\{0,1\}^\star$, if $S \triangleq \{j |\ \mathsf{Blt}_j \neq \bot \wedge \mathsf{VerifyBallot}(\mathsf{Pk}, j, \mathsf{Blt}_j) = \bot\}$ 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 $\mathsf{Blt}'_j = \bot$ 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 $m_{i_1}, \ldots, m_{i_k} \in \mathcal{M} \cup \{\bot\}$, all $\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N \in \{0,1\}^\star \cup \{\bot\}$ 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 $S \triangleq \{j |\ \mathsf{Blt}_j \neq \bot \wedge \mathsf{VerifyBallot}(\mathsf{Pk}, j, \mathsf{Blt}_j) = \bot\}$ 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 $\mathsf{Blt}'_j = \bot$ otherwise, it holds that: if there exist $y \in \{0,1\}^\star \cup \{\bot\}$ 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 $i_1, \ldots, i_k$.

Note that the difference between condition (2) of verifiability and condition (2) of weak verifiability lies in the fact that, in the latter, y cannot equal $\bot$, whereas, in the former, the condition has to hold even for $y = \bot$. In this work, we use the terms verifiability and full verifiability interchangeably to differentiate them from weak verifiability.

Privacy We define *privacy* in the style of indistinguishability-based security. Privacy for a $(N, \mathcal{M}, \Sigma, F)$-eVote

$$\mathsf{EVOTE} \triangleq (\mathsf{Setup}, \mathsf{Cast}, \mathsf{VerifyBallot}, \mathsf{EvalTally}, \mathsf{VerifyTally})$$

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.

The advantage of adversary $\mathcal{A}$ in the above game is defined as

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

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

- 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 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 Cast algorithm.)
- Challenge phase. The challenger does the following. For all $j \in [N]$, if $j \in S$, then 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

Definition 4 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 λ in the above game.

Definition 5 An EVOTE for parameters $(N, \mathcal{M}, \Sigma, F)$ is *weakly private* or wIND-Secure if the advantage of all non-uniform PPT adversaries $\mathcal{A}$ is at most negligible in λ in a game $\mathsf{WeakPriv}_{\mathcal{A}}^{N,\mathcal{M},\Sigma,F,\mathsf{EVOTE}}(1^\lambda)$ identical to the one above except that $\mathcal{A}$ is required to output an empty set S, i.e., $\mathcal{A}$ cannot submit dishonest ballots.

Remark 1 We make some remarks on the previous definitions.

- Our definitions suppose that algorithm $\mathsf{VerifyBallot}$ is run on input each ballot before running algorithm $\mathsf{VerifyTally}$. The ballots that are input to $\mathsf{VerifyTally}$ are replaced by $\perp$ if they were not accepted by $\mathsf{VerifyBallot}$. Another possibility would be to let $\mathsf{VerifyTally}$ do this task itself.
- We require that $\mathsf{VerifyBallot}$ and $\mathsf{VerifyTally}$ be deterministic algorithms. Alternatively, they can be defined as PPT, but then definitions of weak verifiability and 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.
- Both condition (2) of verifiability and condition (2) of weak verifiability lie in some sense between correctness and verifiability as they state a requirement about honest voters.
- In our weakly verifiable construction in Section 4, algorithm $\mathsf{VerifyBallot}$ could be completely discarded because it accepts any ballot. Both for the sake of generality (there could exist some weakly verifiable eVote that makes a non-trivial use of $\mathsf{VerifyBallot}$) and to avoid overburdening the presentation, we use the same syntax for weakly verifiable eVotes and for verifiable eVotes.
- For the necessity of condition (2) of correctness, we refer the reader to the discussion in Section 1.4.
- As shown in [BCG+15], the definition of "Benaloh" (recall that we restate it using modern terminology) is subject to attacks when instantiated with specific tally functions like the majority. Nonetheless, ours is strengthened to withstand such attacks. This is done by adding the 3-rd winning condition.

3 Building Blocks

Our constructions use perfectly binding commitment schemes, (one-message) non-interactive witness-indistinguishable proof systems with perfect soundness for NP [GOS06] (see also [FLS90,DN00,DN00,BOV03,BP15]) and IND-CPA public key encryption with perfect correctness and unique secret key. In this section, we recall the definitions of those primitives.

Definition 6 [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^λ, it outputs public key Pk and decryption key Sk.
- $\mathsf{Encrypt}(m, \mathsf{Pk})$: On input message m and the public key, it outputs ciphertext Ct.
- $\mathsf{Decrypt}(\mathsf{Ct}, \mathsf{Sk})$: On input ciphertext Ct and the decryption key, 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 Ct output by $\mathsf{Encrypt}(\mathsf{Pk}, m)$, $\mathsf{Decrypt}(\mathsf{Ct}, \mathsf{Sk}) = m$ must hold. *Unique secret key* requires that, for all Pk, there exists at most *one* Sk such that $(\mathsf{Pk}, \mathsf{Sk}) \in \mathsf{Setup}(1^\lambda)$.

The Decision Linear Encryption scheme [BBS04] fulfills those properties. It is secure under the Decision Linear Assumption [BBS04]. We recall them next.

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 $\mathcal{G}$.[BBS04]) *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:*

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

The Decision Linear assumption states that the advantage of $\mathcal{A}$ is negligible in λ. Boneh et al. *[BBS04] provide a bilinear group generator $\mathcal{G}$ for which such assumption is conjectured to hold.*

Decision Linear (DLIN) Encryption Scheme. Consider the following PKE scheme described by a setup algorithm Setup, an encryption algorithm $\mathsf{Encrypt}$ and a decryption algorithm $\mathsf{Decrypt}$.

$\mathsf{Setup}(1^\lambda)$**:** 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 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 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 [BBS04] for details) and it is easy to verify that it fulfills the unique secret key property.

Definition 7 [(Perfectly binding) Commitment Schemes] A commitment scheme Com is a PPT algorithm that takes as input a string x and randomness $r \in \{0,1\}^k$ and outputs $\mathsf{com} \leftarrow \mathsf{Com}(x; r)$. 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), for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\nu(\cdot)$ such that: $|\Pr_{r \in \{0,1\}^k}[\,\mathcal{A}(\mathsf{Com}(x_0; r)) = 1\,] - \Pr_{r \in \{0,1\}^k}[\,\mathcal{A}(\mathsf{Com}(x_1; r)) = 1)\,]| \leq \nu(k)$.

(one-message) NIWI proof systems. Next, we define (one-message) non-interactive witness indistinguishability (NIWI) proof systems [GOS06]. Groth *et al.* [GOS06] construct such NIWIs for all languages in NP, and in particular for $\mathsf{CircuitSat}$.

Definition 8 [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 π. Verify receives as input a statement x and a proof π 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 of statements $\{x_k\}_{k>0}$ 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 9 [(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, in short) 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.

4 Our Weakly Verifiable eVote

In this section, we present our weakly verifiable eVote EVOTE. This eVote fulfills the wIND-Security and weak verifiability properties.

Definition 10 [EVOTE] Let $\mathsf{NIWI}^{\mathsf{dec}} = (\mathsf{Prove}^{\mathsf{dec}}, \mathsf{Verify}^{\mathsf{dec}})$ be a NIWI proof system for the relation $\mathsf{R}^{\mathsf{dec}}$, which we specify later. Let $\mathcal{E} = (\mathcal{E}.\mathsf{Setup}, \mathcal{E}.\mathsf{Encrypt}, \mathcal{E}.\mathsf{Decrypt})$ be a PKE scheme with perfect correctness and unique secret key (see Def. 6).

We define as follows an $(N, \mathcal{M}, \Sigma, F)$-eVote

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

- $\mathsf{Setup}(1^\lambda)$: on input the security parameter in unary, do the following.
  1. For all $l \in [3]$, run $(\mathcal{E}.\mathsf{Pk}_l, \mathcal{E}.\mathsf{Sk}_l) = \mathcal{E}.\mathsf{Setup}(1^\lambda; s_l)$ with randomness s_l.
  2. Output $\mathsf{Pk} \stackrel{\triangle}{=} (\mathcal{E}.\mathsf{Pk}_1, \ldots, \mathcal{E}.\mathsf{Pk}_3)$ and $\mathsf{Sk} \stackrel{\triangle}{=} (\mathcal{E}.\mathsf{Sk}_1, \mathcal{E}.\mathsf{Sk}_2, s_1, s_2)$.[10]
- $\mathsf{Cast}(\mathsf{Pk}, j, v)$: on input the public key Pk, the voter index $j \in [N]$, and a vote v, do the following.
  1. For all $l \in [3]$, compute $\mathsf{Ct}_{j,l} \leftarrow \mathcal{E}.\mathsf{Encrypt}(\mathcal{E}.\mathsf{Pk}_l, v)$.
  2. Output $\mathsf{Blt}_j \stackrel{\triangle}{=} (\mathsf{Ct}_{j,1}, \ldots, \mathsf{Ct}_{j,3})$.
- $\mathsf{VerifyBallot}(\mathsf{Pk}, j, \mathsf{Blt})$: on input the public key Pk, the voter index $j \in [N]$, and a ballot Blt, output OK (i.e., accept any ballot, even invalid ones).
- $\mathsf{EvalTally}(\mathsf{Pk}, \mathsf{Sk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N)$: on input the public key Pk, the secret key Sk, and a tuple of N strings $(\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N)$ that consists of either ballots cast by a voter or the special symbol $\perp$, do the following.
  1. For all $j \in [N], l \in [2]$,

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

  2. For all $l \in [2]$, compute $y_l = F(m_{1,l}, \ldots, m_{N,l})$.
  3. If $y_1 = y_2$, then set $y = y_1$, else set $y = \perp$.

[10] Actually, 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 Sk.

4. Consider the following relation $\mathsf{R}^{\mathsf{dec}}$ in Fig. 4. Henceforth, 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). Set the statement

$$x \triangleq (\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, \mathcal{E}.\mathsf{Pk}_1, \ldots, \mathcal{E}.\mathsf{Pk}_3, y)$$

and the witness

$$w \triangleq (\mathcal{E}.\mathsf{Sk}_1, \mathcal{E}.\mathsf{Sk}_2, s_1, s_2, i_1 \triangleq 1, i_2 \triangleq 2)$$

and compute a proof $\gamma \leftarrow \mathsf{Prove}^{\mathsf{dec}}(x, w)$.

5. Output (y, γ).

– $\mathsf{VerifyTally}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y, \gamma)$: on input the public key Pk, a tuple of N strings that can be either ballots cast by a voter or the special symbol $\perp$, a tally y and a proof γ, if $\gamma = \perp$ output $\perp$, else set

$$x \triangleq (\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, \mathcal{E}.\mathsf{Pk}_1, \ldots, \mathcal{E}.\mathsf{Pk}_3, y)$$

output $\mathsf{Verify}^{\mathsf{dec}}(x, \gamma)$.

Henceforth, for simplicity we omit the parameters of the scheme and we write just EVOTE.

4.1 Correctness and Weak Verifiability of the Construction

Correctness. The (perfect) correctness of EVOTE follows from the perfect correctness of $\mathcal{E}$ and the perfect completeness of $\mathsf{NIWI}^{\mathsf{dec}}$.

Weak verifiability.

Theorem 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 with unique secret key (cf. Def. 6) and $\mathsf{NIWI}^{\mathsf{dec}}$ is a (one-message) NIWI (cf. Def. 9) for the relation $\mathsf{R}^{\mathsf{dec}}$, then $\mathsf{EVOTE}^{N,\mathcal{M},\Sigma,F,\mathcal{E},\mathsf{NIWI}^{\mathsf{dec}}}$ satisfies the weak verifiability property (cf. Def. 3).

Proof. First, we prove that condition (1) of verifiability is satisfied. Since algorithm $\mathsf{VerifyBallot}$ accepts any ballot, even invalid ones, we have to prove that for all $\mathsf{Pk} \in \{0,1\}^\star$, all $\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N \in \{0,1\}^\star \cup \{\perp\}$, there exist $m_1, \ldots, m_N \in \mathcal{M} \cup \{\perp\}$ such that for all $y \neq \perp$ and all γ 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 Pk and $\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N$ be arbitrary strings. First, we prove the following claim.

Claim 1 Given Pk and $(\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N)$, for every two pairs (y_0, γ_0) and (y_1, γ_1) such that $y_0, y_1 \neq \perp$, 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 $y_0 = y_1$.

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

Instance: $x \stackrel{\triangle}{=} (\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, \mathcal{E}.\mathsf{Pk}_1, \ldots, \mathcal{E}.\mathsf{Pk}_3, y)$. (Recall that a ballot is set to $\perp$ if the corresponding voter did not cast her vote.)

Witness: $w \stackrel{\triangle}{=} (\mathcal{E}.\mathsf{Sk}'_1, \mathcal{E}.\mathsf{Sk}'_2, s_1, s_2, i_1, i_2)$, 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 $l \in [2]$, $y = F(m_1^l, \ldots, m_N^l)$ 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 Ct_{j,i_l} in Blt_j to $m_j^{i_l} \in \mathcal{M}$; and for all $l \in [2]$, $m_j^l = \perp$ if either $\mathsf{Blt}_j = \perp$ or $\mathcal{E}.\mathsf{Sk}_{i_l}$ decrypts 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 (a) and (c) are not actually conditions that have to be checked but are steps needed to define (note the use of "$\stackrel{\triangle}{=}$") the variables $\mathcal{E}.\mathsf{Pk}_{i_l}$'s, $\mathcal{E}.\mathsf{Sk}_{i_l}$'s and $m_j^{i_l}$'s that are used in the checks (b) and (d).

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

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

(d) $(y = \perp) \vee$ (for all $l \in [2]$, $y = F(m_1^{i_l}, \ldots, m_N^{i_l})$).
(e) $i_1 \neq i_2$.
(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.)

Fig. 4. Relation $\mathsf{R}^{\mathsf{dec}}$

Let $y_0, \gamma_0, y_1, \gamma_1$ be arbitrary strings in $\{0,1\}^\star \cup \{\bot\}$ such that $y_0, y_1 \neq \bot$. 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 γ_b is computed on input some witness $(\mathcal{E}.\mathsf{Sk}_1'^b, \mathcal{E}.\mathsf{Sk}_2'^b, s_1^b, s_2^b, i_1^b, i_2^b)$.

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

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

1. $i^\star = i_1^0 = i_2^1$. For all $b \in \{0,1\}$, let $(m_1^{i^\star,b}, \ldots, m_N^{i^\star,b})$ be the messages guaranteed by condition (c) of relation $\mathsf{R}^{\mathsf{dec}}$ for proof γ_b. Condition (i) for proof γ_0 (resp. γ_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 (b) and (c) for proof γ_0 (resp. γ_1) imply that for all $j \in [N]$, either $m_j^{i^\star,0} = \bot$ 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 $m_j^{i^\star,1} = \bot$ 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 $j \in [N]$, $m_j^{i^\star,0} = m_j^{i^\star,1} \in \mathcal{M} \cup \{\bot\}$. Now, condition (d) for proof γ_0 (resp. γ_1) implies that either $y_0 = F(m_1^{i_1^0,0}, \ldots, m_N^{i_1^0,0})$ or $y_0 = \bot$ (resp. either $y_1 = F(m_1^{i_2^1,1}, \ldots, m_N^{i_2^1,1})$ or $y_1 = \bot$) and, as by hypothesis $y_0, y_1 \neq \bot$, it holds that $y_0 = y_1$. (Here, the "weakness" of EVOTE arises, i.e., it cannot be proven (fully) verifiable because it could occur that, for example, $y_0 \neq y_1, y_0 = \bot$.)
2. $i^\star = i_2^0 = i_1^1$. 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.
3. $i^\star = i_1^0 = i_1^1$. This case is identical to the first one, except that we replace i_2^1 with i_1^1.
4. $i^\star = i_2^0 = i_2^1$. This case is identical to the first one, except that we replace i_1^0 with i_2^0.

In all cases, we have that, 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 $y_0 = y_1$. 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, γ) such that $y \neq \bot$, 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, γ), if $\mathsf{VerifyTally}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y, \gamma) = \mathsf{OK}$, there exist messages $m_1, \ldots, m_N \in \mathcal{M} \cup \{\bot\}$ such that $y = F(m_1, \ldots, m_N)$ (2).

Now, we have two mutually exclusive cases.

- For all (y, γ) such that $y \neq \bot$, $\mathsf{VerifyTally}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y, \gamma) = \bot$. Then, letting $m_1, \ldots, m_N$ in the statement of the theorem be arbitrary messages in $\mathcal{M} \cup \{\bot\}$, the statement is verified with respect to Pk and $\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N$.
- There exists (y', γ) such that $y' \neq \bot$ and $\mathsf{VerifyTally}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y', \gamma) = \mathsf{OK}$. In this case, (2) implies that there exist $m_1', \ldots, m_N' \in \mathcal{M} \cup \{\bot\}$

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, γ) such that $y \neq \perp$, if $\mathsf{VerifyTally}(\mathsf{Pk}, \mathsf{Blt}_1, \mathsf{Blt}_N, y, \gamma) = \mathsf{OK}$ then (by (1)) $y = y^\star =$ (by (4)) $= F(m'_1, \ldots, m'_N)$.
Then, for $m_1 \stackrel{\triangle}{=} m'_1, \ldots, m_N \stackrel{\triangle}{=} m'_N$, the statement of condition (1) of weak verifiability is verified with respect to Pk and $\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N$.

In both cases, for $m_1 \stackrel{\triangle}{=} m'_1, \ldots, m_N \stackrel{\triangle}{=} m'_N$, the statement of condition (1) of weak verifiability is verified with respect to Pk and $\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N$.

As Pk and $\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N$ are arbitrary strings, the statement of condition (1) of weak verifiability is proven.

It is also easy to check that condition (2) of weak 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). 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. 1).

4.2 Weak Privacy of the Construction

Theorem 2 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. 6) and $\mathsf{NIWI}^{\mathsf{dec}}$ is a (one-message) NIWI (cf. Def. 9) for the relation $\mathsf{R}^{\mathsf{dec}}$, then $\mathsf{EVOTE}^{N,\mathcal{M},\Sigma,F,\mathcal{E},\mathsf{NIWI}^{\mathsf{dec}}}$ is wIND-Secure (cf. Def. 5).

Proof. Let $\mathcal{A}$ be a non-uniform PPT adversary against the wIND-Security property of EVOTE. We prove that $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{EVOTE},\mathsf{WeakPriv}}(1^\lambda) \leq \nu(1^\lambda)$ for some negligible function $\nu(\lambda)$.

We prove that by means of a series of hybrid experiments. We refer the reader to Table 1.5 for a pictorial explanation of the experiments, which are explained in Section 1.5. In the table, for simplicity, we omit the indices k for the experiments H_2^k's,H_4^k's,H_6^k's presented below. Therefore, hybrid experiment H_2 (resp. H_4, H_6) in the table corresponds to hybrid experiment H_2^N (resp. H_4^N, H_6^N) below.

Hybrid H_1. Experiment H_1 is equal to the experiment $\mathsf{WeakPriv}_{\mathcal{A}}^{N,\mathcal{M},\Sigma,F,\mathsf{EVOTE}}$ except that the challenger sets $b \stackrel{\triangle}{=} 0$.

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 $j = 1, \ldots, k$, the challenger computes $\mathsf{Ct}_{k,3}$ on input $m_{1,k}$. Note that H_2^0 is identical to H_1.

Claim 2 For all $k = 1, \ldots, N$, the advantage of $\mathcal{A}$ in distinguishing H_2^{k-1} from H_2^k is negligible.

Proof. Suppose toward a contradiction that $\mathcal{A}$ has instead non-negligible advantage $\epsilon(\lambda)$. 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 pk and sets $\mathsf{Pk}_3 \triangleq \mathsf{pk}$. For $l \in [2]$, $\mathcal{B}$ runs $\mathcal{E}.\mathsf{Setup}$ to compute $(\mathcal{E}.\mathsf{Pk}_l, \mathcal{E}.\mathsf{Sk}_l)$ and runs $\mathcal{A}$ on input $\mathsf{Pk} \triangleq (\mathcal{E}.\mathsf{Pk}_1, \mathcal{E}.\mathsf{Pk}_2, \mathcal{E}.\mathsf{Pk}_3)$.

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

$\mathcal{B}$ computes $\mathsf{Blt}_k \triangleq (\mathsf{Ct}_{k,1}, \mathsf{Ct}_{k,2}, \mathsf{ct}^\star)$ by encrypting $m_{0,j}$ in $\mathsf{Ct}_{k,1}$ and $\mathsf{Ct}_{k,2}$. $\mathcal{B}$ can compute the ballots Blt_j for all $j \in [N], j \neq k$ exactly as the challenger in both experiments would do.

$\mathcal{B}$ computes y as in the previous experiment (i.e., by running $\mathsf{EvalTally}$ on input $(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N)$) and uses the 2 secret keys $(\mathcal{E}.\mathsf{Sk}_1, \mathcal{E}.\mathsf{Sk}_2)$ to compute a proof γ exactly as the challenger in both experiments would do. $\mathcal{B}$ restarts $\mathcal{A}$ on input the computed ballots along with (y, γ) and returns the output of $\mathcal{A}$.

It is easy to see that, if $\mathsf{ct}^\star$ is an encryption of $m_{0,k}$, then $\mathcal{B}$ simulates experiment H_2^{k-1}, and, if $\mathsf{ct}^\star$ is an encryption of $m_{1,k}$, then $\mathcal{B}$ simulates experiment H_2^k. Therefore, $\mathcal{B}$ has probability $\epsilon(\lambda)$ 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 γ on input a witness that contains indices $(1, 3)$ and secret keys $\mathsf{Sk}_1, \mathsf{Sk}_3$ (precisely, with the randomness used to compute those secret keys, but henceforth, for simplicity, we omit this detail).

Claim 3 The advantage of $\mathcal{A}$ in distinguishing H_2^N from H_3 is negligible.

Proof. This follows straightforwardly from the WI property of $\mathsf{NIWI}^{\mathsf{dec}}$. We note 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 $(\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, \mathcal{E}.\mathsf{Pk}_1, \ldots, \mathcal{E}.\mathsf{Pk}_3, y)$.

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 $j = 1, \ldots, k$, the challenger computes $\mathsf{Ct}_{k,2}$ on input $m_{1,k}$. Note that H_4^0 is identical to H_3.

Claim 4 For all $k = 1, \ldots, N$, the advantage of $\mathcal{A}$ in distinguishing H_4^{k-1} from H_4^k is negligible.

Proof. The proof is identical to the one for Claim 2 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 γ on input a witness that contains indices $(2,3)$ and secret keys $\mathsf{Sk}_2, \mathsf{Sk}_3$.

Claim 5 The advantage of $\mathcal{A}$ in distinguishing H_4^N from H_5 is negligible.

Proof. This follows straightforwardly from the WI property of $\mathsf{NIWI}^{\mathsf{dec}}$. We note 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 $(\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, \mathcal{E}.\mathsf{Pk}_1, \ldots, \mathcal{E}.\mathsf{Pk}_3, y)$.

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 $j = 1, \ldots, k$, the challenger computes $\mathsf{Ct}_{k,1}$ on input $m_{1,k}$. Note that H_6^0 is identical to H_5.

Claim 6 For all $k = 1, \ldots, N$, the advantage of $\mathcal{A}$ in distinguishing H_6^{k-1} from H_6^k is negligible.

Proof. The proof is identical to the one for Claim 2 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 computes the proof γ on input a witness that contains indices $(1,2)$ and secret keys $(\mathsf{Sk}_1, \mathsf{Sk}_2)$.

Claim 7 The advantage of $\mathcal{A}$ in distinguishing H_6^N from H_7 is negligible.

Proof. This follows straightforwardly from the WI property of $\mathsf{NIWI}^{\mathsf{dec}}$. We note 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 $(\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, \mathcal{E}.\mathsf{Pk}_1, \ldots, \mathcal{E}.\mathsf{Pk}_3, y)$.

Experiment H_1 (resp. H_7) is identical to experiment $\mathsf{WeakPriv}_{\mathcal{A}}^{N,\mathcal{M},\Sigma,F,\mathsf{EVOTE}}$ except that the challenger sets $b = 0$ (resp. $b = 1$). Hence $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{EVOTE},\mathsf{WeakPriv}}(1^\lambda)$ equals at most the sum of the advantages of $\mathcal{A}$ in distinguishing the previous hybrids. Since N is a constant, such advantage is negligible and the theorem is proven.

Corollary 3 If the Decision Linear assumption (see Section 3) holds, then there exists a weakly verifiable eVote.

Proof. Boneh *et al.* [BBS04] show the existence of a PKE scheme with perfect correctness and unique secret key that fulfills the IND-CPA property under the Decision Linear assumption. Groth *et al.* [GOS06] show the existence of (one-message) NIWI proofs with perfect soundness for all languages in NP that is secure under the Decision Linear assumption. Then, because Theorem 1 and Theorem 2 are proven, the corollary follows.

5 Our (Fully) Verifiable eVote

In this Section, we present an eVote scheme $\mathsf{EVOTE}_{\mathsf{full}}$ that is IND-Secure and (fully) verifiable.

Definition 11 [$\mathsf{EVOTE}_{\mathsf{full}}$] Let $\mathcal{E} = (\mathcal{E}.\mathsf{Setup}, \mathcal{E}.\mathsf{Encrypt}, \mathcal{E}.\mathsf{Decrypt})$ be a PKE scheme with perfect correctness and unique secret key (see Def. 6). Let Com be a perfectly binding commitment scheme. Let $\mathsf{NIWI}^{\mathsf{dec,full}} = (\mathsf{Prove}^{\mathsf{dec,full}}, \mathsf{Verify}^{\mathsf{dec,full}})$ and $\mathsf{NIWI}^{\mathsf{enc,full}} = (\mathsf{Prove}^{\mathsf{enc,full}}, \mathsf{Verify}^{\mathsf{enc,full}})$ be two NIWI proof systems for the relations $\mathsf{R}^{\mathsf{dec,full}}$ and $\mathsf{R}^{\mathsf{enc,full}}$, which we specify later.

We define as follows an $(N, \mathcal{M}, \Sigma, F)$-eVote

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

- $\mathsf{Setup}_{\mathsf{full}}(1^\lambda)$: 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 $\mathsf{Pk} \stackrel{\triangle}{=} (\mathcal{E}.\mathsf{Pk}_1, \ldots, \mathcal{E}.\mathsf{Pk}_3, Z)$ and $\mathsf{Sk} \stackrel{\triangle}{=} (\mathcal{E}.\mathsf{Sk}_1, \mathcal{E}.\mathsf{Sk}_2, s_1, s_2, r)$.[11]
- $\mathsf{Cast}_{\mathsf{full}}(\mathsf{Pk}, j, v)$: on input the public key 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 following relation $\mathsf{R}^{\mathsf{enc,full}}$ in Fig. 5. Run $\mathsf{Prove}^{\mathsf{enc,full}}$ on input the statement $(j, \mathsf{Ct}_1, \ldots, \mathsf{Ct}_3, \mathcal{E}.\mathsf{Pk}_1, \ldots, \mathcal{E}.\mathsf{Pk}_3, Z)$ and the witness $(v, r_1, \ldots, r_3)$ to compute a proof π_j. Output $\mathsf{Blt}_j \stackrel{\triangle}{=} (\mathsf{Ct}_{j,1}, \ldots, \mathsf{Ct}_{j,3}, \pi_j)$.
- $\mathsf{VerifyBallot}_{\mathsf{full}}(\mathsf{Pk}, j, \mathsf{Blt})$: on input the public key Pk, the voter index $j \in [N]$, and a ballot Blt, output $\mathsf{Verify}^{\mathsf{enc,full}}((j, \mathsf{Ct}_1, \ldots, \mathsf{Ct}_3, \mathcal{E}.\mathsf{Pk}_1, \ldots, \mathcal{E}.\mathsf{Pk}_3, Z), \pi)$.
- $\mathsf{EvalTally}_{\mathsf{full}}(\mathsf{Pk}, \mathsf{Sk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N)$: on input the public key Pk, the secret key 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 $\mathsf{VerifyBallot}_{\mathsf{full}}(\mathsf{Pk}, j, \mathsf{Blt}_j) = \perp$, 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 $y_l = F(m_{1,l}, \ldots, m_{N,l})$.

[11] Actually, 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 Sk.

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

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

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

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

(a) **Real mode.** All 3 ciphertexts $(\mathsf{Ct}_1, \ldots, \mathsf{Ct}_3)$ 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

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

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

4. If $y_1 = y_2$ then set $y = y_1$, else set $y = \perp$.[12]
5. Consider the following relation $\mathsf{R}^{\mathsf{dec,full}}$ in Fig. 6. (The relation $\mathsf{R}^{\mathsf{dec,full}}$ is identical to the relation $\mathsf{R}^{\mathsf{dec}}$ that is used in our weakly verifiable eVote. The only difference is that the ballots in the statement of $\mathsf{R}^{\mathsf{dec,full}}$ are replaced by $\perp$ if they are not accepted by the ballot verification algorithm. Henceforth, 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 $\mathsf{Prove}^{\mathsf{dec,full}}$ on input the statement $(\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, \mathcal{E}.\mathsf{Pk}_1, \ldots, \mathcal{E}.\mathsf{Pk}_3, y)$ 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 γ.
6. Output (y, γ).

– $\mathsf{VerifyTally}_{\mathsf{full}}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y, \gamma)$: on input the public key Pk, N strings that can be either ballots cast by a voter or the special symbol $\perp$, a tally y and a proof γ of tally correctness, do the following. If $y = \perp$ and all Blt_j's are equal to $\perp$, output OK. If $y = \perp$ but not all Blt_j's are equal to $\perp$, output $\perp$. Otherwise output the decision of $\mathsf{Verify}^{\mathsf{dec,full}}((\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, \mathcal{E}.\mathsf{Pk}_1, \ldots, \mathcal{E}.\mathsf{Pk}_3, y), \gamma)$, after having replaced Blt_j's with $\perp$ when $\mathsf{VerifyBallot}_{\mathsf{full}}(\mathsf{Pk}, j, \mathsf{Blt}_j) = \perp$. Precisely, the algorithm does the following:
1. For all $j \in [N]$, if $\mathsf{VerifyBallot}_{\mathsf{full}}(\mathsf{Pk}, j, \mathsf{Blt}_j) = \perp$, set $\mathsf{Blt}_j = \perp$.
2. If $y \neq \perp$, then output $\mathsf{Verify}^{\mathsf{dec,full}}((\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, \mathcal{E}.\mathsf{Pk}_1, \ldots, \mathcal{E}.\mathsf{Pk}_3, y), \gamma)$.
3. If $y = \perp$, then, if for all $j \in [N], \mathsf{Blt}_j = \perp$, output OK, else output $\perp$.

[12] 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.

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

Instance: $x \stackrel{\triangle}{=} (\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, \mathcal{E}.\mathsf{Pk}_1, \ldots, \mathcal{E}.\mathsf{Pk}_3, y)$. (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: $w \stackrel{\triangle}{=} (\mathcal{E}.\mathsf{Sk}'_1, \mathcal{E}.\mathsf{Sk}'_2, s_1, s_2, i_1, i_2)$, 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).

$\mathsf{R}^{\mathsf{dec,full}}(x, w) = 1$ 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 $l \in [2]$, $y = F(m_1^l, \ldots, m_N^l)$ 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 Ct_{j,i_l} in Blt_j to $m_j^{i_l} \in \mathcal{M}$; and for all $l \in [2]$, $m_j^l = \perp$ if either $\mathsf{Blt}_j = \perp$ or $\mathcal{E}.\mathsf{Sk}_{i_l}$ decrypts Ct_{j,i_l} to a string $\notin \mathcal{M}$.

Precisely, $\mathsf{R}^{\mathsf{dec,full}}(x, w) = 1$ if and only if the following conditions hold. In the following, items (a) and (c) are not actually conditions that have to be checked but are steps needed to define (note the use of "$\stackrel{\triangle}{=}$") the variables $\mathcal{E}.\mathsf{Pk}_{i_l}$'s, $\mathcal{E}.\mathsf{Sk}_{i_l}$'s and $m_j^{i_l}$'s that are used in the checks (b) and (d).

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

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

(d) $(y = \perp) \vee$ (for all $l \in [2]$, $y = F(m_1^{i_l}, \ldots, m_N^{i_l})$).
(e) $i_1 \neq i_2$.
(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.)

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

Henceforth, for simplicity we omit the parameters of the scheme and we just write $\mathsf{EVOTE}_{\mathsf{full}}$.

5.1 Correctness and (Full) Verifiability of the Construction

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

(Full) verifiability.

Theorem 4 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. 6), Com is a PPT algorithm, and $\mathsf{NIWI}^{\mathsf{dec,full}}$ and $\mathsf{NIWI}^{\mathsf{enc,full}}$ are (one-message) NIWIs (cf. Def. 9), for the relations $\mathsf{R}^{\mathsf{dec,full}}$ and $\mathsf{R}^{\mathsf{enc,full}}$ respectively, then $\mathsf{EVOTE}^{N,\mathcal{M},\Sigma,F,\mathcal{E},\mathsf{Com},\mathsf{NIWI}^{\mathsf{enc,full}},\mathsf{NIWI}^{\mathsf{dec,full}}}_{\mathsf{full}}$ satisfies the (full) 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 $\mathsf{VerifyBallot}_{\mathsf{full}}(\mathsf{Pk}, j, \mathsf{Blt}_j) = \mathsf{OK}$, there exist $m_1, \ldots, m_N \in \mathcal{M} \cup \{\perp\}$ such that, for all $y, \gamma \in \{0,1\}^\star$, if $\mathsf{VerifyTally}_{\mathsf{full}}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y, \gamma) = 1$ then $y = F(m_1, \ldots, m_N)$. Henceforth, w.l.o.g, we let 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 $\mathsf{VerifyBallot}_{\mathsf{full}}(\mathsf{Pk}, j, \mathsf{Blt}_j) = \mathsf{OK}$.

First, we prove the following claim.

Claim 8 Given Pk and $(\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N)$, for every two pairs (y_0, γ_0) and (y_1, γ_1), if $\mathsf{VerifyTally}_{\mathsf{full}}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y_0, \gamma_0) = \mathsf{VerifyTally}_{\mathsf{full}}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y_1, \gamma_1) = \mathsf{OK}$ then $y_0 = y_1$.

For every (y_0, γ_0) and (y_1, γ_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, γ), it holds that (A) if $\mathsf{Blt}_1 = \cdots = \mathsf{Blt}_N = \perp$, then $\mathsf{VerifyTally}_{\mathsf{full}}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y, \gamma) = \mathsf{OK}$ if and only if $y = \perp$ and (B) if,

for some $j \in [N], \mathsf{Blt}_j \neq \perp$, then $\mathsf{VerifyTally}_{\mathsf{full}}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, \perp, \gamma) = \perp$. We now have two cases.

(a) $\mathsf{Blt}_1 = \cdots = \mathsf{Blt}_N = \perp$. Then we have that $\mathsf{VerifyTally}_{\mathsf{full}}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y_1, \gamma_1) = \mathsf{OK}$ 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 $\mathsf{VerifyTally}_{\mathsf{full}}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y_0, \gamma_0) = \perp$, which contradicts the fact that (y_0, γ_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 $\mathsf{VerifyTally}_{\mathsf{full}}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y_0, \gamma_0) = \mathsf{VerifyTally}_{\mathsf{full}}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y_1, \gamma_1) = \mathsf{OK}$. The perfect soundness of $\mathsf{NIWI}^{\mathsf{dec,full}}$ implies that, for all $b \in \{0,1\}$, the proof γ_b is computed on input some witness $(\mathcal{E}.\mathsf{Sk}_1'^b, \mathcal{E}.\mathsf{Sk}_2'^b, s_1^b, s_2^b, i_1^b, i_2^b)$.

In the following, when we talk about "condition X for proof γ_b", we refer to the items (a)-(e) in the definition of relation $\mathsf{R}^{\mathsf{dec,full}}$.

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 (c) of relation $\mathsf{R}^{\mathsf{dec,full}}$ for proof γ_b. Condition (i) for proof γ_0 (resp. γ_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 (b) and (c) for proof γ_0 (resp. γ_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 (d) for proof γ_0 (resp. γ_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 $\mathsf{VerifyTally}_{\mathsf{full}}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y_0, \gamma_0) = \mathsf{VerifyTally}_{\mathsf{full}}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y_1, \gamma_1) = \mathsf{OK}$ 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, γ) such that $y \neq \perp$, if $\mathsf{VerifyTally}_{\mathsf{full}}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y, \gamma) = \mathsf{OK}$ then $y = y^\star$ (1). Moreover, it is easy to see that, for all (y, γ), if $\mathsf{VerifyTally}_{\mathsf{full}}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y, \gamma) = \mathsf{OK}$, 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, γ) such that $y \neq \perp$, $\mathsf{VerifyTally}_{\mathsf{full}}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y, \gamma) = \perp$. 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 Pk and $\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N$.
– There exists (y', γ) such that $y' \neq \perp$ and $\mathsf{VerifyTally}_{\mathsf{full}}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y', \gamma) = \mathsf{OK}$. 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, γ) such that $y \neq \perp$, if $\mathsf{VerifyTally}_{\mathsf{full}}(\mathsf{Pk}, \mathsf{Blt}_1, \mathsf{Blt}_N, y, \gamma) = \mathsf{OK}$ then (by (1)) $y = y^\star =$ (by (4)) $= F(m'_1, \ldots, m'_N)$.
Then, for $m_1 \stackrel{\triangle}{=} m'_1, \ldots, m_N \stackrel{\triangle}{=} m'_N$, the statement of condition (1) of weak verifiability is verified with respect to Pk and $\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N$.

In both cases, for $m_1 \stackrel{\triangle}{=} m'_1, \ldots, m_N \stackrel{\triangle}{=} m'_N$, the statement of condition (1) of weak verifiability is verified with respect to Pk and $\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N$.

As Pk and $\mathsf{Blt}_1, \ldots, \mathsf{Blt}_N$ are arbitrary strings, the statement of condition (1) of weak verifiability is proven.

It is also easy to check that condition (2) of weak verifiability is satisfied. This follows straightforwardly from the perfect soundness of $\mathsf{NIWI}^{\mathsf{dec,full}}$. Thanks to $\mathsf{NIWI}^{\mathsf{dec,full}}$, 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. 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 Com is not needed, i.e., the theorem holds for any PPT algorithm Com, even insecure ones.

5.2 Privacy of the Construction

Theorem 5 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. 6), Com is a computationally hiding commitment scheme (cf. Def. 7), and $\mathsf{NIWI}^{\mathsf{dec,full}}$ and $\mathsf{NIWI}^{\mathsf{enc,full}}$ are (one-message) NIWIs (cf. Def. 9), respectively, for the relations $\mathsf{R}^{\mathsf{dec,full}}$ and $\mathsf{R}^{\mathsf{enc,full}}$, then $\mathsf{EVOTE}^{N,\mathcal{M},\Sigma,F,\mathcal{E},\mathsf{Com},\mathsf{NIWI}^{\mathsf{enc,full}},\mathsf{NIWI}^{\mathsf{dec,full}}}_{\mathsf{full}}$ is IND-Secure (cf. Def. 4).

Proof. Consider the following experiment $H^Z_{\mathcal{A}}(1^\lambda)$ between a challenger and a non-uniform PPT adversary $\mathcal{A}$ (henceforth, we often omit the parameters).

Experiment H^Z. H^Z is equal to the experiment $\mathsf{Priv}_{\mathcal{A}}^{N,\mathcal{M},\Sigma,F,\mathsf{EVOTE}_{\mathsf{full}}}$ 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 9 The probability P_0 that $\mathcal{A}$ wins the experiment $\mathsf{Priv}_{\mathcal{A}}^{N,\mathcal{M},\Sigma,F,\mathsf{EVOTE}_{\mathsf{full}}}$ 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 Com with non-negligible probability.

$\mathcal{B}$ receives as input a commitment 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 $\mathsf{Priv}_{\mathcal{A}}^{N,\mathcal{M},\Sigma,F,\mathsf{EVOTE}_{\mathsf{full}}}$ 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 com is a commitment to 1, the probability that $\mathcal{B}$ outputs 1 equals the probability that $\mathcal{A}$ wins in $\mathsf{Priv}_{\mathcal{A}}^{N,\mathcal{M},\Sigma,F,\mathsf{EVOTE}_{\mathsf{full}}}$, and if 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 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 $\mathsf{Priv}_{\mathcal{A}}^{N,\mathcal{M},\Sigma,F,\mathsf{EVOTE}_{\mathsf{full}}}$ 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 $B = m_{0,j} = (\mathsf{Ct}_1, \ldots, \mathsf{Ct}_3)$ (suppose that $m_{0,j}$ can be parsed that way),

it holds that $\mathsf{VerifyBallot}_{\mathsf{full}}(\mathsf{Pk}, j, B) = \mathsf{OK}$ 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 10 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 Com with non-negligible probability.

$\mathcal{B}$ receives as input a commitment 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 $\mathsf{Priv}_{\mathcal{A}}^{N,\mathcal{M},\Sigma,F,\mathsf{EVOTE}_{\mathsf{full}}}$ 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 $B = m_{0,j}$, B can be parsed as $(\mathsf{Ct}_1, \ldots, \mathsf{Ct}_3)$ and it holds that $\mathsf{VerifyBallot}_{\mathsf{full}}(\mathsf{Pk}, j, B) = \mathsf{OK}$ 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 com is a commitment to 1, the perfect soundness of $\mathsf{NIWI}^{\mathsf{enc},\mathsf{full}}$ and the definition of relation $\mathsf{R}^{\mathsf{enc},\mathsf{full}}$ guarantee that the conditions above are never satisfied for any $j \in S$. Therefore, if com is a commitment to 1 $\mathcal{B}$ outputs 1 with probability 1.

On the other hand, if 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 ϵ and 1 with probability $1 - \epsilon$. In conclusion, the advantage of $\mathcal{B}$ in breaking the computationally hiding property of Com is $\epsilon(\lambda)$, which contradicts the assumption that the commitment scheme is computationally hiding.

From Claim 9 and Claim 10, we now know that, for some negligible function $\mathsf{negl}(\cdot)$, the following equations hold:

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

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

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

(Here and henceforth, we omit the parameters, but it is meant that the experiments are parameterized by λ 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. The reader could still

refer to Table 1.5 for a pictorial explanation of the hybrid experiments. However, the experiments in the table, though conceptually very similar, correspond to the security reduction for the weakly verifiable eVote. Moreover, 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 11 For all $k = 1, \ldots, N$, $\left|\Pr\left[\, H_2^{k-1} = 1 | \bar{E}^1 \,\right] - \Pr\left[\, H_2^{k} = 1 | \bar{E}^1 \,\right]\right|$ is negligible.

Proof. Suppose toward a contradiction that the difference between such probabilities is 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 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 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 Blt_j exactly as the challenger in both experiments would do. $\mathcal{B}$ computes y using $\mathsf{EvalTally}_{\mathsf{full}}$ and uses the 2 secret keys $\mathcal{E}.\mathsf{Sk}_1, \mathcal{E}.\mathsf{Sk}_2$ to compute a proof γ exactly as the challenger in both experiments would do. $\mathcal{B}$ sends $\mathcal{A}$ the computed ballots along with (y, γ) 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 γ 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 12 $\left|\Pr\left[\, H_2^{N} = 1 | \bar{E}^1 \,\right] - \Pr\left[\, H_3 = 1 | \bar{E}^1 \,\right]\right|$ is negligible.

Proof. The proof follows from the WI property of $\mathsf{NIWI}^{\mathsf{dec,full}}$. 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 $\mathsf{VerifyBallot}_{\mathsf{full}}$ 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 13 For all $k = 1, \ldots, N$, $\left|\Pr\left[\, H_4^{k-1} = 1 | \bar{E}^1 \,\right] - \Pr\left[\, H_4^k = 1 | \bar{E}^1 \,\right]\right|$ is negligible.

Proof. The proof is identical to the one for Claim 11 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 γ on input a witness that contains indices $(2, 3)$ and secret keys $(\mathsf{Sk}_2, \mathsf{Sk}_3)$.

Claim 14 $\left|\Pr\left[\, H_4^N = 1 | \bar{E}^1 \,\right] - \Pr\left[\, H_5 = 1 | \bar{E}^1 \,\right]\right|$ is negligible.

Proof. This follows straightforwardly from the WI property of $\mathsf{NIWI}^{\mathsf{dec,full}}$. 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 $\mathsf{VerifyBallot}_{\mathsf{full}}$ 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 15 For all $k = 1, \ldots, N$, $\left|\Pr\left[\, H_6^{k-1} = 1 | \bar{E}^1 \,\right] - \Pr\left[\, H_6^k = 1 | \bar{E}^1 \,\right]\right|$ is negligible.

Proof. The proof is identical to the one for Claim 11 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 γ on input a witness that contains indices $(1, 2)$ and secret keys $(\mathsf{Sk}_1, \mathsf{Sk}_2)$.

Claim 16 $\left|\Pr\left[\, H_6^N = 1 | \bar{E}^1 \,\right] - \Pr\left[\, H_7 = 0 | \bar{E}^1 \,\right]\right|$ is negligible.

Proof. The proof follows straightforwardly from the WI property of $\mathsf{NIWI}^{\mathsf{dec,full}}$. 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 $\mathsf{VerifyBallot}_{\mathsf{full}}$ 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. Finally, Claim 9 and equations 2,3 and 4 imply that $\Pr[\mathsf{Priv} = 1] \leq \nu$ for some negligible function ν and the theorem is proven.

Corollary 6 If the Decision Linear assumption (see Section 3) holds, then there exists a (fully) verifiable eVote.

Proof. Boneh *et al.* [BBS04] 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.* [GOS06] 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 4 and Theorem 5 are proven, the corollary follows.

6 Instantiation of $\mathsf{EVOTE}_{\mathsf{full}}$

In this section, we describe an instantiation of our fully-verifiable e-voting scheme $\mathsf{EVOTE}_{\mathsf{full}}$. In Section 6.1, we describe the algorithms of our efficient instantiation without describing the details of the one-message NIWI proofs for the relations $\mathsf{R}^{\mathsf{enc,full}}$ and $\mathsf{R}^{\mathsf{dec,full}}$.

The next subsections describe how we construct efficient one-message NIWI proofs for the relations $\mathsf{R}^{\mathsf{enc,full}}$ and $\mathsf{R}^{\mathsf{dec,full}}$. First, in Section 6.2, 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 6.3, 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 6.4, 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. In Section 6.5, we describe a concrete instantiation of the relations for $\mathsf{R}^{\mathsf{enc,full}}$ and $\mathsf{R}^{\mathsf{dec,full}}$ over bilinear groups, which can be used in our construction in Section 6.1.

Finally, in Section 6.6, we analyze the efficiency of our instantiation of $\mathsf{EVOTE}_{\mathsf{full}}$.

$$\Pr\left[H^Z = 1|\bar{E}^1\right] =$$

$$\Pr\left[H^Z = 1|\bar{E}^1 \wedge b = 0\right]\Pr\left[b = 0\right] + \Pr\left[H^Z = 1|\bar{E}^1 \wedge b = 1\right]\Pr\left[b = 1\right] =$$

$$= 1/2 \cdot \left(\Pr\left[H^Z = 1|\bar{E}^1 \wedge b = 0\right] + \Pr\left[H^Z = 1|\bar{E}^1 \wedge b = 1\right]\right) =$$

(since H_1 is identically distributed to H^Z with bit $b = 0$ and H_7 to H^Z with $b = 1$)

$$= 1/2 \cdot \left(\Pr\left[H_1 = 1|\bar{E}^1\right] + \Pr\left[H_7 = 1|\bar{E}^1\right]\right) =$$

$$= 1/2 + 1/2 \cdot \left(\Pr\left[H_1 = 1|\bar{E}^1\right] - \Pr\left[H_7 = 0|\bar{E}^1\right]\right) =$$

(since H_1 (resp. H_3, H_5) is identically distributed to H_2^0 (resp. H_4^0, H_6^0))

$$= 1/2 + 1/2 \cdot (\sum_{k=0}^{N-1} (\Pr\left[H_2^k = 1|\bar{E}^1\right] - \Pr\left[H_2^{k+1} = 1|\bar{E}^1\right]) + (\Pr\left[H_2^N = 1|\bar{E}^1\right] - \Pr\left[H_4^0 = 1|\bar{E}^1\right]) +$$

$$\sum_{k=0}^{N-1} (\Pr\left[H_4^k = 1|\bar{E}^1\right] - \Pr\left[H_4^{k+1} = 1|\bar{E}^1\right])(\Pr\left[H_4^N = 1|\bar{E}^1\right] - \Pr\left[H_6^0 = 1|\bar{E}^1\right]) +$$

$$\sum_{k=0}^{N-1} (\Pr\left[H_6^k = 1|\bar{E}^1\right] - \Pr\left[H_6^{k+1} = 1|\bar{E}^1\right])(\Pr\left[H_6^N = 1|\bar{E}^1\right] - \Pr\left[H_7 = 0|\bar{E}^1\right])) \leq$$

$$1 \leq /2 + 1/2 \cdot |(\sum_{k=0}^{N-1} (\Pr\left[H_2^k = 1|\bar{E}^1\right] - \Pr\left[H_2^{k+1} = 1|\bar{E}^1\right]) + (\Pr\left[H_2^N = 1|\bar{E}^1\right] - \Pr\left[H_4^0 = 1|\bar{E}^1\right]) +$$

$$\sum_{k=0}^{N-1} (\Pr\left[H_4^k = 1|\bar{E}^1\right] - \Pr\left[H_4^{k+1} = 1|\bar{E}^1\right])(\Pr\left[H_4^N = 1|\bar{E}^1\right] - \Pr\left[H_6^0 = 1|\bar{E}^1\right]) +$$

$$\sum_{k=0}^{N-1} (\Pr\left[H_6^k = 1|\bar{E}^1\right] - \Pr\left[H_6^{k+1} = 1|\bar{E}^1\right])(\Pr\left[H_6^N = 1|\bar{E}^1\right] - \Pr\left[H_7 = 0|\bar{E}^1\right]))| \leq$$

(by the triangle inequality)

$$\leq 1/2 + 1/2 \cdot (\sum_{k=0}^{N-1} |\Pr\left[H_2^k = 1|\bar{E}^1\right] - \Pr\left[H_2^{k+1} = 1|\bar{E}^1\right]| + |(\Pr\left[H_2^N = 1|\bar{E}^1\right] - \Pr\left[H_4^0 = 1|\bar{E}^1\right])| +$$

$$\sum_{k=0}^{N-1} |\Pr\left[H_4^k = 1|\bar{E}^1\right] - \Pr\left[H_4^{k+1} = 1|\bar{E}^1\right]||\Pr\left[H_4^N = 1|\bar{E}^1\right] - \Pr\left[H_6^0 = 1|\bar{E}^1\right]| +$$

$$\sum_{k=0}^{N-1} |\Pr\left[H_6^k = 1|\bar{E}^1\right] - \Pr\left[H_6^{k+1} = 1|\bar{E}^1\right]||\Pr\left[H_6^N = 1|\bar{E}^1\right] - \Pr\left[H_7 = 0|\bar{E}^1\right]|) \leq$$

(by Claims 11 - 16)

$$\leq 3k \cdot \mathsf{negl}, \text{where } \mathsf{negl} \text{ is the sum of the negligible functions guaranteed by Claims 11 - 16.} \quad (4)$$

Fig. 7. Equation 4

6.1 Algorithms of our Efficient Instantiation of $\mathsf{EVOTE}_{\mathsf{full}}$

$\mathsf{EVOTE}_{\mathsf{full}}$ 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. 6). We use the DLIN encryption scheme, which we recall in Section 3, in order to instantiate the public-key encryption scheme.

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

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

In the following, we describe the algorithms of our instantiation of $\mathsf{EVOTE}_{\mathsf{full}}$. In our general scheme in Section 5, the setup algorithm 3 public keys and a commitment value Z. In this instantiation, the setup algorithm outputs an additional public key Pk_4 to be used as the parameters of the commitment scheme used for computing Z.

- $\mathsf{Setup}_{\mathsf{full}}(1^\lambda)$: 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 $\mathsf{Sk} \leftarrow (\mathcal{E}.\mathsf{Sk}_1, \mathcal{E}.\mathsf{Sk}_2)$.
- $\mathsf{Cast}_{\mathsf{full}}(\mathsf{Pk}, j, v)$: on input the public key 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 π_j for the relation $\mathsf{R}^{\mathsf{enc,full}}$. We describe a construction for one-message NIWI in Section 6.4 and we describe an instantiation of the relation $\mathsf{R}^{\mathsf{enc,full}}$ over bilinear groups in Section 6.5.
  4. Output $\mathsf{Blt}_j = (\mathsf{Ct}_{j,1}, \ldots, \mathsf{Ct}_{j,3}, \pi_j)$.
- $\mathsf{VerifyBallot}_{\mathsf{full}}(\mathsf{Pk}, j, \mathsf{Blt})$: on input the public key Pk, the voter index $j \in [N]$, and a ballot Blt, verify the proof π_j in the ballot by using the verification algorithm of our one-message NIWI construction in Section 6.4 for the relation $\mathsf{R}^{\mathsf{enc,full}}$ over bilinear groups in Section 6.5. Output the verification result.
- $\mathsf{EvalTally}_{\mathsf{full}}(\mathsf{Pk}, \mathsf{Sk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N)$: on input the public key Pk, the secret key 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 $\mathsf{VerifyBallot}_{\mathsf{full}}(\mathsf{Pk}, j, \mathsf{Blt}_j) = \perp$, 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 γ for the relation $\mathsf{R}^{\mathsf{dec,full}}$. We describe a construction for one-message NIWI in Section 6.4 and we describe an instantiation of the relation $\mathsf{R}^{\mathsf{dec,full}}$ over bilinear groups in Section 6.5.
6. Output (y, γ).

– $\mathsf{VerifyTally}_{\mathsf{full}}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y, \gamma)$: 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 γ 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 $\mathsf{R}^{\mathsf{dec,full}}$ 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 $\mathsf{VerifyBallot}_{\mathsf{full}}(\mathsf{Pk}, j, \mathsf{Blt}_j) = \perp$, set $\mathsf{Blt}_j = \perp$.
3. If $y \neq \perp$, then verify the proof γ by using the verification algorithm of our one-message NIWI construction in Section 6.4 for the relation $\mathsf{R}^{\mathsf{dec,full}}$ over bilinear groups in Section 6.5. Output the verification result.
4. If $y = \perp$, then, if for all $j \in [N], \mathsf{Blt}_j = \perp$, output OK, else output $\perp$.

6.2 NIWI Proofs Secure Under a Trusted Setup

The following definitions are taken from [GS08]. 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 12 [NIWI proof system in the CRS model] A NIWI proof system in the CRS model for the relation R consists of four algorithms $\mathsf{NIWISetup}$, $\mathsf{NIWIKeygen}$, $\mathsf{NIWIProve}$ and $\mathsf{NIWIVerify}$. 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 π. $\mathsf{NIWIVerify}(gk, crs, x, \pi)$ outputs 1 if π 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 $\mathsf{NIWIVerify}$ accepts the proofs computed by algorithm $\mathsf{NIWIProve}$. 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 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:

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

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

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

6.3 Groth-Sahai NIWI Proofs

Groth and Sahai [GS08] 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 and in the "hiding" setting 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 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.* [GSW10].

It is possible to compute a proof of the disjunction of two or more pairing product equations [Gro06]. 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 Δ_0 and Δ_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 (Δ_0, Δ_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 δ_0 and δ_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 δ_0 and δ_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 $C = (f^{s_1} u^{s_3}, h^{s_2} v^{s_3}, Xg^{s_1+s_2} w^{s_3})$, which is a DLIN encryption of X. A perfectly hiding commitment is $C = (f^{s_1} u^{s_3}, h^{s_2} v^{s_3}, Xg^{-s_3} g^{s_1+s_2} w^{s_3})$.

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 $(u^x f^{s_1}, v^x h^{s_2}, w^x g^{r_1+r_2})$.

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 $\mathsf{R}^{\mathsf{enc,full}}$ and $\mathsf{R}^{\mathsf{dec,full}}$ described in Section 6.6.

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 3×3. 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.

6.4 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 [GOS12]. In [GOS12], 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 $\mathsf{CircuitSat}$ in [GOS12] can be used to construct NIWI proofs for the relations $\mathsf{R}^{\mathsf{enc,full}}$ and $\mathsf{R}^{\mathsf{dec,full}}$. However, the resulting NIWI proofs are inefficient.

In [GS08], Groth and Sahai provide NIWI and NIZK proofs for the satisfiability of equations over bilinear groups. The NIWI proofs in [GS08] use a CRS as trust assumption, and therefore we cannot use them directly to construct NIWI proofs for the relations $\mathsf{R}^{\mathsf{enc,full}}$ and $\mathsf{R}^{\mathsf{dec,full}}$. However, we observe that the NIWI and NIZK proofs in [GS08] 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 [GOS12], 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 [GOS12], the prover computes two NIZK proofs instead of NIWI proofs, but we observe that witness-indistinguishability is sufficient.

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.* [GOS12], 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 [GOS12], and so the property of verifiable correlated key generation also holds for Groth-Sahai proofs.[13]

We describe now the algorithms $(\mathsf{Prove}^R, \mathsf{Verify}^R)$ defined in Section 3 of a one-message NIWI proof system (*without* CRS) for satisfiability of equations over bilinear groups. Prove^R and Verify^R receive as input (gk, sk) output by $\mathsf{NIWISetup}(1^k)$. In our instantiation of $\mathsf{EVOTE}_{\mathsf{full}}$ in Section 6.1, gk is replaced by the pairing group setup Γ 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 π, 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 [GOS12] 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.

[13] We note that the perfectly binding commitment key (resp. perfectly hiding commitment key) in [GOS12] is the perfectly hiding commitment key (resp. perfectly binding commitment key) in [GS08]. The reason is the way the commitments are computed in [GOS12] and in [GS08]. Nevertheless, the arguments to show that the commitment keys satisfy the property of verifiable correlated key generation given in [GOS12] also hold for [GS08].

Completeness and soundness. 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.

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.
3. Hybrid 3. The third hybrid proceeds as the first, except that π_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 π_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.

6.5 Groth-Sahai NIWI Proofs for $\mathsf{R}^{\mathsf{enc,full}}$ and $\mathsf{R}^{\mathsf{dec,full}}$

Groth-Sahai NIWI Proofs for $\mathsf{R}^{\mathsf{enc,full}}$. We describe an instantiation of the relation $\mathsf{R}^{\mathsf{enc,full}}$ 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 $\mathsf{R}^{\mathsf{enc,full}}$.

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.

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

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 $\mathsf{R}^{\mathsf{enc,full}}$, 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 $\mathcal{E}.\mathsf{Pk}_l = (\Gamma, g_l, f_l, h_l)$.[14] Consider the following relation R_2.

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

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 $\mathsf{R}^{\mathsf{enc,full}}$, 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

[14] 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 3, all g_l's values correspond to the same group element g that is contained in Γ.

$\{1, g\} \in \mathbb{G}$. Consider the following relation R_3.

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

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 Δ_0 and Δ_1. These variables are used to compute an OR proof of pairing product equations as described in Section 6.3. 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, α_1, β_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 8 our instantiation of relation $\mathsf{R}^{\mathsf{enc,full}}$ over bilinear groups. $\mathsf{R}^{\mathsf{enc,full}}$ 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 $1 \in G$. As described in Section 6.1, 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 Δ_2 and Δ_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 δ_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 $\mathsf{R}^{\mathsf{dec,full}}$. We describe an instantiation of the relation $\mathsf{R}^{\mathsf{dec,full}}$ 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 $\mathsf{R}^{\mathsf{dec,full}}$.

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 $\mathcal{E}.\mathsf{Sk} \leftarrow$

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,full}} = \{(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. 8. Relation $\mathsf{R}^{\mathsf{enc,full}}$

(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.

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

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 $\mathsf{R}^{\mathsf{dec,full}}$, 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.

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

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 Δ_0 and Δ_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, α_1, β_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 $\mathsf{R}^{\mathsf{dec,full}}$, 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. Let $L_x = \{1, 2\}$, $L_y = \{1, 3\}$ and $L_z = \{2, 3\}$.

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

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 Δ_x, Δ_y and Δ_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 $\mathsf{R}^{\mathsf{dec,full}}$, 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 m_j is a vote. We show in Figure 9 our instantiation of relation $\mathsf{R}^{\mathsf{dec,full}}$ over bilinear groups. We introduce the variables δ_t for $t \in \{x, y, z\}$, which equal the tally y if $\Delta_t \neq 1$.

6.6 Efficiency of Our Instantiation of $\mathsf{EVOTE}_{\mathsf{full}}$

In this section, we analyze the communication and computational cost of our instantiation of $\mathsf{EVOTE}_{\mathsf{full}}$ described in Section 6.1. 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. In Table 2, we summarize the communication cost, and, in Table 3, we summarize the computational cost. We note that the tables show the size of one ballot

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,full}} = \{(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. 9. Relation $\mathsf{R}^{\mathsf{dec,full}}$

Blt_j and the execution time of $\mathsf{Cast}_{\mathsf{full}}$ and $\mathsf{VerifyBallot}_{\mathsf{full}}$ on input one ballot. Additionally, the execution time of $\mathsf{EvalTally}_{\mathsf{full}}$ and $\mathsf{VerifyTally}_{\mathsf{full}}$ does not include the N executions of $\mathsf{VerifyBallot}_{\mathsf{full}}$ required to verify the ballots, where N is the number of ballots that do not equal $\perp$.

Communication Cost of $\mathsf{EVOTE}_{\mathsf{full}}$. Algorithm $\mathsf{Setup}_{\mathsf{full}}(1^\lambda)$ 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 Γ 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 Γ), 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 Pk is $12 \cdot |\mathbb{G}|$ bits. The size of the secret key Sk is $4 \cdot |\mathbb{Z}_p|$ bits.

Algorithm $\mathsf{Cast}_{\mathsf{full}}(\mathsf{Pk}, j, v)$ 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 π_j consists of two correlated common references strings and two Groth-Sahai NIWI proofs for the relation $\mathsf{R}^{\mathsf{enc,full}}$. The size of the two correlated common reference strings is $7 \cdot |\mathbb{G}|$. The witness of $\mathsf{R}^{\mathsf{enc,full}}$ consists of $m = 34$ elements of $\mathbb{G}$. $\mathsf{R}^{\mathsf{enc,full}}$ 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 π_j is $661 \cdot |\mathbb{G}|$. The size of a ballot is $670 \cdot |\mathbb{G}|$.

Algorithm $\mathsf{EvalTally}_{\mathsf{full}}(\mathsf{Pk}, \mathsf{Sk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N)$ outputs the tally y and the proof γ. The proof γ consists of two correlated common references strings and two Groth-Sahai NIWI proofs for the relation $\mathsf{R}^{\mathsf{dec,full}}$. The size of the two correlated common reference strings is $7 \cdot |\mathbb{G}|$. The witness of $\mathsf{R}^{\mathsf{dec,full}}$ consists of $m = 6 + 90 \cdot N$ elements of $\mathbb{G}$, where N is the number of ballots that do not equal $\perp$. $\mathsf{R}^{\mathsf{dec,full}}$ 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 γ is $(139 + 1836 \cdot N) \cdot |\mathbb{G}|$.

Algorithms $\mathsf{VerifyBallot}_{\mathsf{full}}(\mathsf{Pk}, j, \mathsf{Blt})$ and $\mathsf{VerifyTally}_{\mathsf{full}}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y, \gamma)$ output one bit.

Table 2. Communication Cost of $\mathsf{EVOTE}_{\mathsf{full}}$

Public key Pk	$12 \cdot \lvert\mathbb{G}\rvert$
Secret key Sk	$4 \cdot \lvert\mathbb{Z}_p\rvert$
Ballot Blt_j	$670 \cdot \lvert\mathbb{G}\rvert$
Proof γ	$(139 + 1836 \cdot N) \cdot \lvert\mathbb{G}\rvert$

Computational Cost of $\mathsf{EVOTE}_{\mathsf{full}}$. Algorithm $\mathsf{Setup}_{\mathsf{full}}(1^\lambda)$ 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 $\mathsf{Cast}_{\mathsf{full}}(\mathsf{Pk}, j, v)$ 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 π_j consists of two correlated common references strings and two Groth-Sahai NIWI proofs for the relation $\mathsf{R}^{\mathsf{enc,full}}$. The computation time of two correlated common reference strings is $5 \cdot |\mathsf{exp}|$. The witness of $\mathsf{R}^{\mathsf{enc,full}}$ 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 $102 \cdot |\mathsf{mul}|$. $\mathsf{R}^{\mathsf{enc,full}}$ 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 Blt_j is $14 \cdot |\mathsf{exp}| + 1224 \cdot |\mathsf{mul}|$.

Algorithm $\mathsf{VerifyBallot}_{\mathsf{full}}(\mathsf{Pk}, j, \mathsf{Blt})$ verifies the proof π_j for relation $\mathsf{R}^{\mathsf{enc,full}}$ in the ballot. This implies verifying two Groth-Sahai NIWI proofs for relation $\mathsf{R}^{\mathsf{enc,full}}$. $\mathsf{R}^{\mathsf{enc,full}}$ 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 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 π_j is $2130 \cdot |\mathsf{map}|$.

Algorithm $\mathsf{EvalTally}_{\mathsf{full}}(\mathsf{Pk}, \mathsf{Sk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N)$ computes the tally y and a proof γ for relation $\mathsf{R}^{\mathsf{dec,full}}$. The computation of y requires decrypting $2N$ ciphertexts. Each ciphertext involves 2 exponentiations. The computation of γ involves computing two correlated common references strings and two Groth-Sahai NIWI proofs for the relation $\mathsf{R}^{\mathsf{dec,full}}$. The computation time of two correlated common reference strings is $5 \cdot |\mathsf{exp}|$. The witness of $\mathsf{R}^{\mathsf{dec,full}}$ 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 $(18 + 270 \cdot N) \cdot |\mathsf{mul}|$. $\mathsf{R}^{\mathsf{dec,full}}$ 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 γ is $5 \cdot |\mathsf{exp}| + (111 + 1818 \cdot N) \cdot |\mathsf{mul}|$ and the computation time of $\mathsf{EvalTally}_{\mathsf{full}}(\mathsf{Pk}, \mathsf{Sk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N)$ is $(5 + 4 \cdot N) \cdot |\mathsf{exp}| + (222 + 3636 \cdot N) \cdot |\mathsf{mul}|$.

Algorithm $\mathsf{VerifyTally}_{\mathsf{full}}(\mathsf{Pk}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_N, y, \gamma)$ verifies a proof γ. This implies verifying two Groth-Sahai NIWI proofs for relation $\mathsf{R}^{\mathsf{dec,full}}$. $\mathsf{R}^{\mathsf{dec,full}}$ 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 γ is $(414 + 6264 \cdot N) \cdot |\mathsf{map}|$.

Table 3. Computational Cost of $\mathsf{EVOTE}_{\mathsf{full}}$

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

7 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 [Bit14] for a survey).
- **Receipt-freeness.** Perfect verifiability and perfect correctness seem incompatible with receipt-freeness [BT94,SK95,MH96,MN06,DKR09,CCFG16], 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 [CCFG16].
- **Other applications of our techniques.** We think that our techniques could be of wide applicability to other settings. For instance, Camenisch and Shoup [CS03a] put forth the concept of verifiable encryption (that in some sense could be also viewed as a special case of verifiable functional encryption [BGJS16]) 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.

8 Acknowledgments

We thank Saikrishna Badrinarayanan, Aayush Jain, Steve Kremer and Johannes Mueller for suggestions and helpful discussions about verifiability.

Vincenzo Iovino is supported by the Luxembourg National Research Fund (FNR grant no. 7884937). Further, this work is 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.

References

Adi08. Ben Adida. Helios: Web-based open-audit voting. In *USENIX Security Symposium*, volume 17, pages 335–348, 2008.

BBS04. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, August 2004.

BCG+15. David Bernhard, Véronique Cortier, David Galindo, Olivier Pereira, and Bogdan Warinschi. Sok: A comprehensive analysis of game-based ballot privacy definitions. In *2015 IEEE Symposium on Security and Privacy*, pages 499–516. IEEE, 2015.

BDPA11. G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. The Keccak reference, 2011. `http://keccak.noekeon.org/`.

BDSG+13. Nir Bitansky, Dana Dachman-Soled, Sanjam Garg, Abhishek Jain, Yael Tauman Kalai, Adriana López-Alt, and Daniel 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.*, pages 182–201. Springer, 2013.

Ben87. J. Benaloh. *Verifiable secret-ballot elections*. PhD thesis, Yale University, 1987.

BF03. Dan Boneh and Matthew K. Franklin. Identity based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003.

BFM88. Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th Annual ACM Symposium on Theory of Computing*, pages 103–112. ACM Press, May 1988.

BFS16. Mihir Bellare, Georg Fuchsbauer, and Alessandra 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*, pages 777–804, 2016.

BGJS16. Saikrishna Badrinarayanan, Vipul Goyal, Aayush Jain, and Amit 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*, pages 557–587, 2016.

Bit14. Nir Bitansky. *Getting inside the Adversarys Head: New Directions in Non-Black-Box Knowledge Extraction*. PhD thesis, Tel Aviv University, 2014.

BOV03. Boaz Barak, Shien Jin Ong, and Salil P. Vadhan. Derandomization in cryptography. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 299–315. Springer, August 2003.

BP15. Nir Bitansky and Omer Paneth. Zaps and non-interactive witness indistinguishability from indistinguishability obfuscation. In *Theory of Cryptography Conference*, pages 401–427. Springer, 2015.

BR93. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73. ACM Press, November 1993.

BSW11. Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 253–273. Springer, March 2011.

BT94. Josh Cohen Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *26th Annual ACM Symposium on Theory of Computing*, pages 544–553. ACM Press, May 1994.

CCC+09. David Chaum, Richard Carback, Jeremy Clark, Aleksander Essex, Stefan Popoveniuc, Ronald L. Rivest, Peter Y. A. Ryan, Emily Shen, Alan T. Sherman, and Poorvi L. Vora. Scantegrity II: end-to-end verifiability by voters of optical scan elections through confirmation codes. *IEEE Trans. Information Forensics and Security*, 4(4):611–627, 2009.

CCFG16. Pyrros Chaidos, Véronique Cortier, Georg Fuchsbauer, and David 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*, pages 1614–1625, 2016. Full version available at `http://eprint.iacr.org/2015/629`.

CG15. Pyrros Chaidos and Jens 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*, pages 650–670, 2015.

CGGI14. Véronique Cortier, David Galindo, Stéphane Glondu, and Malika Izabachène. Election verifiability for helios under weaker trust assumptions. In Miroslaw Kutylowski and Jaideep Vaidya, editors, *ESORICS 2014: 19th European Symposium on Research in Computer Security, Part II*, volume 8713 of *Lecture Notes in Computer Science*, pages 327–344. Springer, September 2014.

CGH98. Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th Annual ACM Symposium on Theory of Computing*, pages 209–218. ACM Press, May 1998.

CGK+16. Véronique Cortier, David Galindo, Ralf Küsters, Johannes Mueller, and Tomasz 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*, pages 779–798, 2016.

CGS97. Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. In Walter Fumy, editor,

Advances in Cryptology – EUROCRYPT'97, volume 1233 of *Lecture Notes in Computer Science*, pages 103–118. Springer, May 1997.

Cha81. David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

CHK04. Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 207–222. Springer, May 2004.

CKLM13. Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Verifiable elections that scale for free. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013: 16th International Workshop on Theory and Practice in Public Key Cryptography*, volume 7778 of *Lecture Notes in Computer Science*, pages 479–496. Springer, February / March 2013.

CPSV16. Michele Ciampi, Giuseppe Persiano, Luisa Siniscalchi, and Ivan 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*, pages 83–111, 2016.

CS03a. Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 126–144. Springer, August 2003.

CS03b. Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003.

CS10. Veronique Cortier and Ben Smyth. Attacking and fixing helios: An analysis of ballot secrecy. Cryptology ePrint Archive, Report 2010/625, 2010. `http://eprint.iacr.org/2010/625`.

CZZ+15. Nikos Chondros, Bingsheng Zhang, Thomas Zacharias, Panos Diamantopoulos, Stathis Maneas, Christos Patsonakis, Alex Delis, Aggelos Kiayias, and Mema Roussopoulos. A distributed, end-to-end verifiable, internet voting system. *CoRR*, abs/1507.06812, 2015.

DDO+01. Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit 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*, pages 566–598, 2001.

DFN06. Ivan Damgård, Nelly Fazio, and Antonio Nicolosi. Non-interactive zero-knowledge from homomorphic encryption. In Shai Halevi and Tal Rabin, editors, *TCC 2006: 3rd Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 41–59. Springer, March 2006.

DH76. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

DJ01. Ivan Damgård and Mats Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In Kwangjo Kim, editor, *PKC 2001: 4th International Workshop on Theory and Practice in Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 119–136. Springer, February 2001.

DJ03. Ivan Damgård and Mads Jurik. A length-flexible threshold cryptosystem with applications. In Reihaneh Safavi-Naini and Jennifer Seberry, editors, *ACISP 03: 8th Australasian Conference on Information Security and Privacy*, volume 2727 of *Lecture Notes in Computer Science*, pages 350–364. Springer, July 2003.

DKR09. Stéphanie Delaune, Steve Kremer, and Mark Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, 2009.

DMP88. Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Non-interactive zero-knowledge proof systems. In Carl Pomerance, editor, *Advances in Cryptology – CRYPTO'87*, volume 293 of *Lecture Notes in Computer Science*, pages 52–72. Springer, August 1988.

DN00. Cynthia Dwork and Moni Naor. Zaps and their applications. In *41st Annual Symposium on Foundations of Computer Science*, pages 283–293. IEEE Computer Society Press, November 2000.

Fed12. Federal Agency on Technical Regulation and Metrology. Gost r 34.11-2012: Streebog hash function, 2012. `https://www.streebog.net`.

FLS90. Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *31st Annual Symposium on Foundations of Computer Science*, pages 308–317. IEEE Computer Society Press, October 1990.

FS87. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, August 1987.

GGG+14. Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 578–602. Springer, May 2014.

GGH+13. Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual Symposium on Foundations of Computer Science*, pages 40–49. IEEE Computer Society Press, October 2013.

GGHZ16. Sanjam Garg, Craig Gentry, Shai Halevi, and Mark Zhandry. Functional encryption without obfuscation. In Eyal Kushilevitz and Tal Malkin, editors, *Theory of Cryptography: 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part II*, pages 480–511. Springer, 2016.

GH07. Matthew Green and Susan Hohenberger. Blind identity-based encryption and simulatable oblivious transfer. In Kaoru Kurosawa, editor, *Advances in Cryptology – ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 265–282. Springer, December 2007.

GIR16. Rosario Giustolisi, Vincenzo Iovino, and Peter 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.

GK03. Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. In *44th Annual Symposium on Foundations of Computer Science*, pages 102–115. IEEE Computer Society Press, October 2003.

GKP+13. Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 555–564. ACM Press, June 2013.

GM84. Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

GO14. Jens Groth and Rafail Ostrovsky. Cryptography in the multi-string model. *Journal of Cryptology*, 27(3):506–543, July 2014.

Gol01. Oded Goldreich. *Foundations of Cryptography: Basic Techniques*, volume 1. Cambridge University Press, Cambridge, UK, 2001.

GOS06. Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for NIZK. In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 97–111. Springer, August 2006.

GOS12. Jens Groth, Rafail Ostrovsky, and Amit Sahai. New techniques for noninteractive zero-knowledge. *Journal of the ACM (JACM)*, 59(3):11, 2012.

Gro04. Jens Groth. Efficient maximal privacy in boardroom voting and anonymous broadcast. In *International Conference on Financial Cryptography*, pages 90–104. Springer, 2004.

Gro06. Jens 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*, pages 444–459. Springer, 2006.

GS08. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432. Springer, April 2008.

GSW10. Essam Ghadafi, Nigel P. Smart, and Bogdan Warinschi. Groth-Sahai proofs revisited. In Phong Q. Nguyen and David Pointcheval, editors, *PKC 2010: 13th International Conference on Theory and Practice of Public Key Cryptography*, volume 6056 of *Lecture Notes in Computer Science*, pages 177–192. Springer, May 2010.

GVW12. Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 162–179. Springer, August 2012.

HRZ10. Feng Hao, Peter Y. A. Ryan, and Piotr Zielinski. Anonymous voting by two-round public discussion. *IET Information Security*, 4(2):62–67, 2010.

JCJ10. Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In *Towards Trustworthy Elections*, pages 37–63. Springer, 2010.

Jou04. Antoine Joux. A one round protocol for tripartite Diffie-Hellman. *Journal of Cryptology*, 17(4):263–276, September 2004.

Kal06. Yael Tauman Kalai. *Attacks on the Fiat-Shamir paradigm and program obfuscation*. PhD thesis, Massachusetts Institute of Technology, 2006.

KRS10. Steve Kremer, Mark Ryan, and Ben Smyth. Election verifiability in electronic voting protocols. In *European Symposium on Research in Computer Security*, pages 389–404. Springer, 2010.

KSRH12. Dalia Khader, Ben Smyth, Peter Y. A. Ryan, and Feng 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*, pages 285–299, 2012.

KY02. Aggelos Kiayias and Moti Yung. Self-tallying elections and perfect ballot secrecy. In David Naccache and Pascal Paillier, editors, *PKC 2002: 5th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2274 of *Lecture Notes in Computer Science*, pages 141–158. Springer, February 2002.

KZZ15. Aggelos Kiayias, Thomas Zacharias, and Bingsheng 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*, pages 468–498, 2015.

Lin15. Yehuda 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*, pages 93–109, 2015.

Lip05. Helger Lipmaa. Secure electronic voting protocols. In Hossein Bidgoli, editor, *Handbook of Information Security, Volume 2, Information Warfare, Social, Legal, and International Issues and Security Foundations*, pages 647–657. John Wiley & Sons, Inc., 2005. Electronic edition available at `http://kodu.ut.ee/~lipmaa/papers/voting4hb.pdf`.

LP09. Yehuda Lindell and Benny Pinkas. A proof of security of Yao's protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, April 2009.

MH96. Markus Michels and Patrick Horster. Some remarks on a receipt-free and universally verifiable mix-type voting scheme. In Kwangjo Kim and Tsutomu Matsumoto, editors, *Advances in Cryptology – ASIACRYPT'96*, volume 1163 of *Lecture Notes in Computer Science*, pages 125–132. Springer, November 1996.

MN06. Tal Moran and Moni Naor. Receipt-free universally-verifiable voting with everlasting privacy. In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 373–392. Springer, August 2006.

Nao03. Moni Naor. On cryptographic assumptions and challenges (invited talk). In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 96–109. Springer, August 2003.

NY90. Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd Annual ACM Symposium on Theory of Computing*, pages 427–437. ACM Press, May 1990.

Ràf15. Carla 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*, pages 247–276, 2015.

Riv06. Ronald L. Rivest. The threeballot voting system, 2006.

RR06. Brian Randell and Peter Y. A. Ryan. Voting technologies and trust. In *IEEE Security and Privacy*, pages 50–56, 2006.

RRI16. Peter Y. A. Ryan, Peter B. Rønne, and Vincenzo 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*, pages 176–192, 2016.

RS92. Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer, August 1992.

RS06. Peter Y. A. Ryan and S. A. Schneider. Prêt à voter with re-encryption mixes. Technical Report CS-TR-956, University of Newcastle, 2006.

RT09. Peter Y. A. Ryan and Vanessa Teague. Pretty good democracy. In *IN: WORKSHOP ON SECURITY PROTOCOLS*, 2009.

SK95. Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme - a practical solution to the implementation of a voting booth. In Louis C. Guillou and Jean-Jacques Quisquater, editors, *Advances in Cryptology – EUROCRYPT'95*, volume 921 of *Lecture Notes in Computer Science*, pages 393–403. Springer, May 1995.

SS10. Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 10: 17th Conference on Computer and Communications Security*, pages 463–472. ACM Press, October 2010.

Yao86. Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science*, pages 162–167. IEEE Computer Society Press, October 1986.