On the Possibility of Non-Interactive E-Voting in the Public-key Setting

Rosario Giustolisi, Vincenzo Iovino, Peter Rønne

On the Possibility of Non-Interactive E-Voting in the Public-key Setting

Rosario Giustolisi[1], Vincenzo Iovino[2], and Peter B. Rønne[3]

[1] SICS Swedish ICT, fgiustol@gmail.com
[2] University of Luxembourg, vinciovino@gmail.com
[3] INRIA Nancy and University of Luxembourg, peter.roenne@inria.fr

Abstract. In 2010 Hao, Ryan and Zielinski proposed a simple decentralized e-voting protocol that only requires 2 rounds of communication. Thus, for k elections their protocol needs $2k$ rounds of communication.
Observing that the first round of their protocol is aimed to establish the public-keys of the voters, we propose an extension of the protocol as a *non-interactive* e-voting scheme in the public-key setting (NIVS) in which the voters, after having published their public-keys, can use the corresponding secret-keys to participate in an arbitrary number of *one-round* elections.
We first construct a NIVS with a standard tally function where the number of votes for each candidate is counted.
Further, we present constructions for two alternative types of elections. Specifically in the first type (*dead or alive elections*) the tally shows if *at least one* voter cast a vote for the candidate. In the second one (*elections by unanimity*), the tally shows if *all* voters cast a vote for the candidate.
Our constructions are based on bilinear groups of prime order.
As definitional contribution we provide formal computational definitions for privacy and verifiability of NIVSs. We conclude by showing intriguing relations between our results, secure computation, electronic exams and conference management systems.

Keywords: e-voting, bilinear maps, secure computation, electronic exams, conference management systems.

Table of Contents

1 Introduction

Background. In 2010 Hao, Ryan and Zielinski [HRZ10] (see also [KSRH12]) designed a simple decentralized e-voting protocol that only needs 2 rounds of communication and is (publicly) verifiable. Their protocol for n participants can be summarized as follows. Let us assume that a trusted authority sets up a Diffie-Hellman [DH76] group $\mathbb{G}$ of prime order p with generator g. In the first round, each voter j chooses a secret element $x_j \leftarrow \mathbb{Z}_p$ and forwards g^{x_j} to the public bulletin-board. Now, each voter j computes the value $g^{y_j} \triangleq g^{\sum_{k<j} x_k - \sum_{k>j} x_k}$ and in the second round sends her ballot $\mathsf{Blt}_j \triangleq g^{v_j} g^{x_j y_j}$, where $v_j \in \{0,1\}$ is her vote.
From the values Blt_j's the tally can be computed as the product, in fact it is easy to see that $\prod_{j\in[n]} g^{x_j y_j} = 1$ and thus $r \triangleq \prod_{j\in[n]} \mathsf{Blt}_j = g^{\sum v_j}$.
Assuming that the result is small it can be computed by computing the discrete log of r in base g. The previous explanation is an oversimplification that skips some aspects, like zero-knowledge proofs for verifiability, that we will take into consideration later.

1.1 Multiple non-interactive elections in the PK setting

The Public-key Setting. The first round of the protocol outlined above can be viewed as the publication of the public-key (PK, henceforth) of the users. That is, we can imagine the element g^{x_j} as the PK of user j and x_j as her secret-key (SK, henceforth). After establishing these pairs of PKs/SKs the voter can cast her vote *non-interactively* (i.e., in a *single* round of interaction).
Note also that non-interactive e-voting is provable impossible to achieve without the PK setting because it clashes with any reasonable notion of privacy. In fact, if it were possible to compute the result of a 0/1 election[4] from a tuple S of n ballots computed in a non-interactive way, then it would be possible to perform the following attack: discard the first $n-1$ ballots in S and replace them with another tuple of ballots that all encode the vote for 0, and compute the tally to learn the vote of the n-th voter in S.
We thus raise the following question:

> In a PK setting, can we achieve a protocol that allows the voters to participate in an *unbounded* number of elections? That is, after the users make public their PKs and retaining for them the corresponding SKs, is it possible for them to engage in an unbounded number of *one-round* voting protocols?

The protocol of Hao *et al.* fails to satisfy this property. In fact, even if in their scheme we consider the first round as the establishment of the PKs/SKs, if the voters make two non-interactive elections then the privacy is completely broken. The reason is that the ballots are not randomized so that the two ballots belonging to the same voter leak her votes.
We solve the issue by resorting to bilinear maps [BF01,Jou04]. Our new protocol extends the one of Hao *et al.* as follows. First of all, we will associate a unique identifier $\mathsf{id} \in \{0,1\}^\lambda$ to each election. That is, voters will associate an identifier id to their ballots and only ballots for the same identifier (i.e., for the same election) can be put together to compute

[4] In the sequel we will use the terms 0/1 and YES/NO elections as synonyms. This type of election is also known as *referendum*.

the tally.

Let us assume a bilinear instance $\mathcal{I} \triangleq (p, \mathbb{G}, \mathbb{G}_T, \mathbf{e})$ (see Section 2.2) in which $\mathbb{G}$ is a group of prime order p and $\mathbf{e}$ is a bilinear function mapping elements of $\mathbb{G}$ to elements of $\mathbb{G}_T$ satisfying non-degeneracy and bilinearity, and let Hash be a hash function taking as input $\mathcal{I}$, an identifier of an election id and outputs elements of $\mathbb{G}$. In our analysis Hash will be modeled as a Random Oracle (RO, in short) model [BR93]. Our protocol in the PK setting is described next.

All voters randomly choose their secret-key $x_j \leftarrow \mathbb{Z}_p$ and publish their public-key $\mathsf{Pk}_j = g^{x_j}$. Each voter computes a random value $\mathsf{Hash}(\mathcal{I}, \mathsf{id}) \in \mathbb{G}$ to be used in the election associated with identifier id.

In election id each voter j will cast her vote v_j as

$\mathsf{Blt}_j \triangleq \mathbf{e}(g^{y_j}, \mathsf{Hash}(\mathcal{I}, \mathsf{id}))^{x_j} \cdot \mathbf{e}(g^{v_j}, \mathsf{Hash}(\mathcal{I}, \mathsf{id}))$, where g^{y_j} is computed from the PKs g^{x_j}'s exactly as in the Hao *et al.*'s protocol described above. As will be explained below, the ballot is cast with a proof of well-formedness.

If we write $g_{\mathsf{id}} = \mathbf{e}(g, \mathsf{Hash}(\mathcal{I}, \mathsf{id}))$ the ballot can be written as $\mathsf{Blt}_j = g_{\mathsf{id}}^{v_j} g_{\mathsf{id}}^{x_j y_j}$ and the relation to Hao et al.'s approach becomes clear. In the target group of the bilinear map, we have constructed a hash function creating new generators for each election in such a way that the PK for any participant in the new generator can be calculated by the other participants and the SKs stay unchanged.

Privacy game. This new model calls for new security definitions. We define the privacy for non-interactive e-voting schemes in the PK setting (NIVS, in short) by means of the following game.

The challenger computes a pair of PK/SK for each voter and feeds the adversary with the PKs. Then a random bit b is chosen and the adversary can adaptively make an unbounded number of queries to an oracle invoking it with two sets of votes, S_0 and S_1 with the same sum and receiving back the ballots computed with S_b by means of the SKs. At any point the adversary can output its guess b' and it wins the game iff $b' = b$.

A formal definition that also takes in account an adversary that corrupts a set of voters seeing their SKs is given in Section 2.1.

We will prove the following theorem.

Theorem 1 If the Bilinear Decision Diffie-Hellman Assumption [Boy08] defined in Section 2.2 holds, then in the RO model no non-uniform PPT adversary can break the privacy (see Definition 2) of the scheme of Section 3 with non-negligible probability.

The proof is given in Section 3.1. Note that the privacy definition does not capture e.g. vote copying attacks. In fact, it implicitly assumes a perfect synchronous broadcast channel. We postpone a stronger ballot privacy definition for future work.

Verifiability. As a further definitional contribution we provide a formal definition for verifiability. Verifiability for NIVSs is somewhat different from schemes with trusted authorities. For example everybody, also third parties, can perform the tally. Further we will think of being in a setting where the ballots and proofs are cast using authenticated channel using the PK structure. Alternative signatures can be added. This prevents attacks where an adversary votes on behalf of another voter. Intuitively verifiability should then guarantee the ability of verifying that a voter cast a ballot according to the vote rules and that the tally has ideal functionality. First of all, let us analyze how a well-formed ballot

look like.
We expect that a well-formed ballot give consistent results with other honestly computed ballots. That is, a ballot Blt should uniquely determine a vote v that, along with any other set of valid ballots, results in a consistent computation.
Our definition of verifiability given in Section 2.1 is divided in two parts (that have to hold together). The first part states that a ballot uniquely determines a vote v such that for any other set of ballots corresponding to another vector of votes $\boldsymbol{v}$ the output of the algorithm that computes the tally will be equal either to the output of the functionality with inputs v and $\boldsymbol{v}$ or to an error $\perp$.
The second part states that there exists an algorithm $\mathsf{VerifyBallot}$ whose aim is to verify the well-formedness of a ballot Blt such that if the verification passes for Blt then for any other set B of honestly computed ballots the result of the tally with respect to the set of ballots $B \cup \{\mathsf{Blt}\}$ will not result in an error $\perp$.
In order to guarantee verifiability of the above sketched NIVS, like in Hao *et al.*, we add proofs of well-formedness of the ballot. Specifically we add a proof that the vote in the ballot is 0 or 1 using the Cramer *et al.* technique adapted to the bilinear setting. We discuss this in Section 2.3. Note that unlike Hao *et al.* we do not add proofs of knowledge to the public-keys as in our work we do not address malleability or copying attacks.
We stress that the proof of well-formedness of the ballot is sufficient to satisfy our notion of verifiability. However, it is easy to see that one can also add proofs of knowledge to the PKs to prevent stronger attacks not taken in account in our model.

We note that this protocol is not fair, e.g. the last to cast a ballot can compute the result before casting her own vote. As explained in [KSRH12] this can be mitigated by an extra commitment round. Also the protocol is not robust, we cannot tally if someone fails to vote. This was also considered in [KSRH12] and in this event it is enough to run an extra round to recover the result.

1.2 Beyond YES/NO elections

The drawback of the previous scheme is that it only supports YES/NO elections. Hao *et al.* showed how to extend their basic scheme to handle multiple candidates by using parallel repetition, i.e., making the voter to cast a ciphertext for any candidate. They mention possible improvements with better communication efficiency at the cost of having a very expensive tallying procedure. However, none of their solutions support multiple elections.

Multiple candidates. The techniques of Hao *et al.* for handling multiple candidates also extend to our context in a natural way. However, we can do even better. The point is that in the same way that we can construct independent generators for each election, we can also inside each election construct independent generators for each candidate $g_{\mathsf{id},a} = \mathbf{e}(g, \mathsf{Hash}(\mathcal{I}, \mathsf{id}, a))$ where $a \in \{1, \ldots, c\}$ and we have c candidates. Voter j now casts c ballots $\mathsf{Blt}_{j,a} = g_{\mathsf{id},a}^{v_{j,a}} g_{\mathsf{id},a}^{x_j y_j}$ where $v_{j,a}$ is 1 for the chosen candidate and 0 otherwise. The voter gives a single zero-knowledge proof of this. Namely, she gives an OR-proof that the El Gamal encryption $(\prod_a g_{\mathsf{id},a}^{x_j}, \prod_a \mathsf{Blt}_{j,a})$ is an encryption of one of the elements in the set $\{g_{\mathsf{id},1}, \ldots, g_{\mathsf{id},c}\}$. Unlike the parallel approach of Hao *et al.* the voter can only cast one vote and we thus have a standard c-candidate election. Where the vote casting part scales roughly with c, the tally part is almost as efficient as in the YES/NO case and the brute force calculation requires maximally n calculations.

However in this work we also present a novel approach to support multiple elections in the PK setting for elections using special tally functions.

Dead or alive elections. In the sequel we consider two new special YES/NO elections and when we say that the voter casts (resp. does not cast) a vote for the candidate we mean that she casts 1 (resp. 0). In the first one, that we call *dead or alive elections* we have 1 candidate and the tally has to compute the predicate $P_{\neq 0}$ that is true iff at least one voter cast a vote for the candidate.
The idea is to change the previous NIVS for YES/NO elections so that if the j-th voter casts a vote for 0 she sets $v_j = 0$, otherwise she sets v_j to a *random* number in $\mathbb{Z}_p$. That is the ballot Blt_j will be defined as $\mathsf{Blt}_j \triangleq \mathbf{e}(g^{y_j}, \mathsf{Hash}(\mathsf{id}, \mathcal{I}))^{x_j} \cdot \mathbf{e}(g^{v_j}, \mathsf{Hash}(\mathsf{id}, \mathcal{I}))$, but with v_j set as described before.

As before, the product of the $\mathbf{e}(g^{y_j}, \mathsf{Hash}(\mathsf{id}, \mathcal{I}))^{x_j}$'s will cancel out when tallying. Hence, only the product of the $\mathbf{e}(g^{v_j}, \mathsf{Hash}(\mathsf{id}, \mathcal{I}))$'s will be left. Therefore, this part will be null if and only if no voter cast a vote for the candidate. Note that here, as the result is Boolean, we do not need to make brute-force computation to extract the result and thus to assume that the number of voters be small.

Elections by unanimity. With a similar technique we can handle *elections by unanimity*, in which the tally has to compute the predicate $P_\forall$ that is true iff all voters cast a vote for the candidate, where a voter can cast a vote for a single candidate.
The idea is similar to the former except that in a ballot for voter j we invert the setting of v_j by choosing $v_j = 0$ if the voter wants to cast a 1 vote or choosing v_j at random in $\mathbb{Z}_p$ if the voter wants to cast a 0 vote. Note that in this case the $\mathbf{e}(g^{v_j}, \mathsf{Hash}(\mathsf{id}, \mathcal{I}))$-part will be null either if and only if *all* voters cast a vote for the candidate.
For both dead or alive elections and elections by unanimity, it would be better (to prevent further attacks not considered in our model) that the votes be cast with a proof of knowledge of v_j, though not adding any proof does not break either the privacy or verifiability as defined.[5]

We mention that both schemes could be extended to support multiple candidates using bilinear groups of composite order but we skip the details.
A drawback of the above constructions (for dead or alive elections and elections by unanimity) is that if an adversary, who is a participating voter, manages to perform a privacy attack on voter j and gets to know v_j, then this adversary can cancel the vote of j by voting $-v_j$. That is, a privacy attack can be turned into an undetectable verifiability attack. In the setting of composite groups we also plan to repair on this. However, we stress that these attacks are not taken in account by our security model but in a future work we will generalize it to withstand stronger attacks.

1.3 Relation to secure computation

Our results relate to secure computation [Yao88,Gol04] of specific functionalities. A recent result of Garg *et al.* [GGHR14] showed the first 2-rounds secure computation protocol in

[5] Precisely, for the verifiability to hold, it is sufficient to check that the ballot be a valid group element, as any group element is in the range of the Cast algorithm.

the CRS model for any functionality.
However, even if we wish to use the protocol of Garg *et al.* to execute k secure evaluations of the functions described in this paper, we would need $2k$ rounds of communication. Instead, using our NIVSs we only need $k + 1$ rounds, one for establishing the PKs and one for each non-interactive secure function evaluation (of the functions supported by our schemes) in the PK setting.
Another related cryptographic notion is Input-Indistinguishable Computation proposed by Micali, Pass and Rosen [MPR06] that shares the indistinguishability-based flavor of NIVS but was implemented with more rounds than ours (though the main focus of the authors was on general functionalities and security under concurrent executions).
It seems that Multi-input Functional Encryption (MIFE, in short) [GGG+14] could be also used to obtain a form of a NIVS in the CRS model (setting the CRS to a token for the desired function). However, this is not straightforward since MIFE would have to be likely combined with signature schemes and, as the indistinguishability-security of MIFE only holds when the two challenge vectors of inputs are not 'splittable' under the functionality,[6] it would offer no security guarantee because there will be exist many values splitting the challenge vectors.[7] A generalization of MIFE studied by Iovino and Żebrowski [IZ15] could be useful in this context.
It is an intriguing research direction to investigate the class of functionalities we can compute in our setting.

1.4 Applications to secure conference management systems and e-exams

Our results are applicable also in the context of secure exams and conference management. In fact, the voters can represent the examiners (or reviewers) who assign a *grade* to a homework (or scientific paper), and the tally corresponds to its evaluation. More specifically, let $[d]$ be the set of possible grades. Our first NIVS for YES/NO elections can be easily extended to support votes (that now will represent grades) in $[d]$ so that the tally divided by the number of examiners would give the average grade for the homework.

Our schemes for dead or alive elections and elections by unanimity could be also useful for the review process of a conference management system.

Our NIVS can be used to get a first evaluation of a paper from the committee members who are involved into the review process. The goal is to preserve anonymity of the review grades also towards the chair, that is, a form of strong blind reviewing. Here, a vote for 1 corresponds to *acceptance* and a vote for 0 to *rejection* (as said before, the scheme can be easily extended to support different grades such as *borderline*). The chair of the conference who is in charge for accepting or rejecting a paper computes the tally to get a first evaluation of the paper without knowning the grades assigned by each reviewers to the paper.

If a paper gets a very high (resp. very low) average grade, then the chair can easily take the final decision, otherwise he may assign "Maybe Accept" (resp. "Maybe Reject") to the

[6] For instance a value z splits two vectors (x_1, x_2) and (y_1, y_2) under a function f if $f(x_1, z) \neq f(y_1, z)$ or $f(z, x_2) \neq f(z, y_2)$. Two vectors are splittable if there exists a value z that splits them.

[7] Precisely, whereas it would be *difficult* to find an input that splits the two challenge vectors under the functionality (as it accounts to forge a signature), such splitting inputs *exist* and thus the security of MIFE is vacuous.

paper. If a paper has been assigned the grade of "Maybe Accept", the chair may call for a dead or alive election to reject the paper iff all committee members will vote rejection. On the other hand, if the paper has been assigned the grade of "Maybe Reject", the chair may call for an election by unanimity to accept the paper iff all committee members agree on acceptance.

Similarly, we expect further applications of our work to secure exams in general.

1.5 Our results in a nutshell

Our contributions can be summarized as follows.

- **A new model.** We introduce the novel concept of non-interactive voting schemes in the PK setting that extends the two-rounds elections of Hao *et al.*. In this model, n voters publish their public-keys retaining the corresponding secret-keys, and each voter using her secret-key can compute her ballot and send it to a public bulletin board. Then, the n ballots can be put together to compute the result of the election.
  Therefore, in this model k elections can be executed with $k+1$ rounds of communications whereas using Hao *et al.*'s schemes would result in $2k$ rounds.
- **Formal definitions.** In Section 2.1 we provide formal definitions for non-interactive voting schemes in the PK setting, in particular for privacy and verifiability, for which a formal treatment was missing.
- **Scheme for YES/NO elections.** In Section 3 we present a non-interactive voting scheme in the PK setting for YES/NO elections (i.e, in which each voter can cast 0 or 1 and the tally computes the sum of all votes) that is provably secure from the Bilinear Decision Diffie-Hellman assumption.
- **Alternative types of elections.** In Section 1.2 we present schemes for alternative types of (YES/NO) elections that could be of independent interest. In particular we can support a *dead or alive election* in which n voters can choose 1 candidate and the result shows for if *at least one* voter cast a vote for him. Another type of election we support is *election by unanimity*, in which the result shows if *all* voters cast a vote for the candidate.
- **Relation to secure computation.** In Section 1.3 we show relations between our results and secure computation.
- **Applications to secure electronic exams and conference systems.** In Section 1.4 we show that our results have direct applications to secure electronic exams and conference management systems.

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 $\{1,\ldots,n\}$, and we shorten *Probabilistic Polynomial-Time* as PPT. If g and A are elements of the same cyclic group, we denote by $\mathbf{dlog}_g A$ the discrete log of A in base g. 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.

2.1 Non-interactive voting scheme in the PK setting

A non-interactive voting scheme in the PK setting (NIVS, in short) is associated with a natural number $n > 0$, the *number of voters*, a set D, the *domain of valid votes*, a set Σ, the *range of possible results*, and a *count function* $F : D^n \to \Sigma$. After that an authority sets-up the public parameters pp, each voter generates a pair of public- and secret- keys. By means of an algorithm Cast and of her own secret-key each voter can cast her vote $v \in D$ generating a ballot Blt and, using the public-keys of all voters, the tally can be publicly computed by means of an algorithm $\mathsf{EvalTally}$. A single ballot can be verified to be the output of the Cast algorithm with input a valid vote $v \in D$ and with respect to a public-key of a voter by means of the algorithm $\mathsf{VerifyBallot}$.

Definition 1 [Non-Interactive Voting Scheme] A (n, D, Σ, F)-*non-interactive voting scheme in the PK setting* NIVS for number of voters n, domain of valid voters D, range of possible results Σ and count function F is a tuple

$\mathsf{NIVS} \triangleq (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Cast}, \mathsf{VerifyBallot}, \mathsf{EvalTally})$ of 5 algorithms with the following syntax:

1. $\mathsf{Setup}(1^\lambda)$, on input the security parameter in unary, outputs *public* parameters pp.
2. $\mathsf{KeyGen}(\mathsf{pp})$, on input the public parameters pp outputs a *public-key* Pk and a *secret-key* Sk.
3. $\mathsf{Cast}(\mathsf{pp}, j, \mathsf{id}, \mathsf{Sk}, (\mathsf{Pk})_{i\in[n]-\{j\}}, v)$, on input the public parameters pp, the secret-key Sk of voter j, the identifier $\mathsf{id} \in \{0,1\}^\lambda$ of the election, the public keys $(\mathsf{Pk}_i)_{i\in[v]-\{j\}}$ of the other voters, and a vote $v \in D$, outputs a *ballot* Blt;
4. $\mathsf{VerifyBallot}(\mathsf{pp}, \mathsf{Pk}, \mathsf{id}, \mathsf{Blt})$, on input the public parameters pp, a public-key Pk of a voter, the identifier $\mathsf{id} \in \{0,1\}^\lambda$ of the election and a ballot Blt, outputs a value in $\{\perp, \mathsf{OK}\}$;
5. $\mathsf{EvalTally}(\mathsf{pp}, \mathsf{Pk}_1, \ldots, \mathsf{Pk}_n, \mathsf{id}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_n)$, on input the public parameters pp, the public-keys of all voters, the identifier $\mathsf{id} \in \{0,1\}^\lambda$ of the election, and the ballots cast by all voter, outputs $y \in \Sigma \cup \{\perp\}$.

Correctness and verifiability. In addition we require the following properties.

1. Correctness or self-tallying. For all $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, for all $(\mathsf{Pk}_1, \mathsf{Sk}_1), \ldots, (\mathsf{Pk}_n, \mathsf{Sk}_n)$ such that for all $i \in [n]$ $(\mathsf{Pk}_i, \mathsf{Sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, all $v_1, \ldots, v_n \in D$, for all identifiers $\mathsf{id} \in \{0,1\}^\lambda$, for all $\mathsf{Blt}_1, \ldots, \mathsf{Blt}_n$ such that for all $i \in [v]$ $\mathsf{Blt}_i \leftarrow \mathsf{Cast}(\mathsf{pp}, j, \mathsf{id}, \mathsf{Sk}, (\mathsf{Pk})_{i\in[n]-\{j\}}, v)$, we have that $\mathsf{EvalTally}(\mathsf{pp}, \mathsf{Pk}_1, \ldots, \mathsf{Pk}_n, \mathsf{id}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_n) = F(v_1, \ldots, v_n)$.
2. Verifiability or dispute-freeness. To not overburden the presentation, we first present a simplified notion of verifiability against one malicious voter and then we will discuss how to extend it to withstand any number of malicious voters. The definition of verifiability against one malicious voter is the following.
   For all except negligible fraction of [8] $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, all $j \in [n]$, all Pk, all Blt, there

[8] In the sequel, we use the expression "all except negligible fraction of..." to mean that the statement holds for all except negligible fraction of the randomness values which the object is computed from. For instance, by "for all except negligible fraction of $(\mathsf{Pk}_i, \mathsf{Sk}_i)_{i\in[n]-\{j\}}$ such that for all $i \in [n] - \{j\}$ $(\mathsf{Pk}_i, \mathsf{Sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$..." we mean that for all except negligible fraction of the randomness values $r \in \{0,1\}^\lambda$, for $(\mathsf{Pk}_i, \mathsf{Sk}_i)_{i\in[n]-\{j\}}$ such that for all $i \in [n]-\{j\}$ $(\mathsf{Pk}_i, \mathsf{Sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp}; r)$...

exists a vote $v \in D$ such that:
for all identifiers $\mathsf{id} \in \{0,1\}^\lambda$, all except negligible fraction of $(\mathsf{Pk}_i, \mathsf{Sk}_i)_{i\in[n]-\{j\}}$ such that for all $i \in [n]-\{j\}$ $(\mathsf{Pk}_i, \mathsf{Sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, and all $v_i \in D$ with $i \in [n]-\{j\}$ and all except negligible fraction of $(\mathsf{Blt}_i)_{i\in[n]-\{j\}}$ satisfying $\mathsf{Blt}_i \leftarrow \mathsf{Cast}(\mathsf{pp}, i, \mathsf{id}, \mathsf{Sk}, (\mathsf{Pk})_{j\in[n]-\{i\}}, v_i)$, it holds that
$\mathsf{EvalTally}(\mathsf{pp}, \mathsf{Pk}_1, \ldots, \mathsf{Pk}_{j-1}, \mathsf{Pk}, \mathsf{Pk}_{j+1}, \ldots, \mathsf{Pk}_n, \mathsf{id}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_{j-1}, \mathsf{Blt}, \mathsf{Blt}_{j+1}, \ldots, \mathsf{Blt}_n)$
outputs either $F(v_1, \ldots, v_{j-1}, v, v_{j+1}, \ldots, v_n)$ or $\perp$.

In addition, we require the following to hold. For all except negligible fraction of $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, for all $j \in [n]$, all Pk, all Blt, if $\mathsf{VerifyBallot}(\mathsf{pp}, \mathsf{Pk}, \mathsf{id}, \mathsf{Blt}) = \mathsf{OK}$ then:
for all identifiers $\mathsf{id} \in \{0,1\}^\lambda$, all except negligible fraction of $(\mathsf{Pk}_i, \mathsf{Sk}_i)_{i\in[n]-\{j\}}$ such that for all $i \in [n]-\{j\}$ $(\mathsf{Pk}_i, \mathsf{Sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, all $v_1, \ldots, v_{n_1} \in D$, all except negligible fraction of $(\mathsf{Blt}_i)_{i\in[n]-\{j\}}$ such that for all $i \in [v]-\{j\}$ $\mathsf{Blt}_i \leftarrow \mathsf{Cast}(\mathsf{pp}, j, \mathsf{id}, \mathsf{Sk}, (\mathsf{Pk})_{i\in[n]-\{j\}}, v)$, it holds that
$\mathsf{EvalTally}(\mathsf{pp}, \mathsf{Pk}_1, \ldots, \mathsf{Pk}_{j-1}, \mathsf{Pk}, \mathsf{Pk}_{j+1}, \ldots, \mathsf{Pk}_n, \mathsf{id}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_{j-1}, \mathsf{Blt}, \mathsf{Blt}_{j+1}, \ldots, \mathsf{Blt}_n) \neq \perp$.
As said, the above definition only takes in account a single malicious voter because we quantify over a single, possibly malicious, voter j, a single, possibly maliciously computed, PK Pk of voter j and a single, possibly maliciously computed, ballot Blt of voter j. By quantifying over all sets of up to n voters and changing it in the obvious way we get the actual definition.

Privacy. Now we formalize the notion of *privacy* (also called *maximal ballot privacy* in Hao *et al.*) in the style of indistinguishability-based security for encryption and related primitives. The privacy for a (n, D, Σ, F)-NIVS $\mathsf{NIVS} \triangleq (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Cast}, \mathsf{VerifyBallot}, \mathsf{EvalTally})$ is formalized by means of the following game $\mathsf{Priv}_{\mathcal{A}}^{n,D,\Sigma,F,\mathsf{NIVS}}$ between an adversary (with access to an oracle) $\mathcal{A} \triangleq (\mathcal{A}_0, \mathcal{A}_1)$ and a *challenger* $\mathcal{C}$.

$\mathsf{Priv}_{\mathcal{A}}^{n,D,\Sigma,F,\mathsf{NIVS}}(1^\lambda)$

- Setup phase. $\mathcal{C}$ generates $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, choose a random bit $b \leftarrow \{0,1\}$ and runs $\mathcal{A}_0$ on input pp;
- Corruption phase. $\mathcal{A}_0$, on input pp, outputs a set $S \subset [n]$ of indices of voters it wants to corrupt.
- Key Generation Phase. For all $i \in [n]$ the challenger generates n pairs of public- and secret- keys $(\mathsf{Pk}_i, \mathsf{Sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, and runs $\mathcal{A}_1^{\mathsf{Vote}(\cdot)}$ on input $(\mathsf{Pk}_i, \mathsf{Sk}_i)_{i\in S}$ and $(\mathsf{Pk}_i)_{i\in[n]-S}$.
- Query phase. The adversary $\mathcal{A}_1$ has access to a stateful oracle Vote. The oracle Vote on input an identifier $\mathsf{id} \in \{0,1\}^\lambda$ and a pair of vectors $\boldsymbol{v}_0 \triangleq (v_{0,1}, \ldots, v_{0,n})$ and $\boldsymbol{v}_1 \triangleq (v_{1,1}, \ldots, v_{1,n})$ outputs the set of ballots $(\mathsf{Cast}(\mathsf{pp}, 1, \mathsf{id}, \mathsf{Sk}_1, (\mathsf{Pk}_i)_{i\in[n]-\{1\}}, v_{b,1}), \ldots, \mathsf{Cast}(\mathsf{pp}, n, \mathsf{id}, \mathsf{Sk}_n, (\mathsf{Pk}_i)_{i\in[n]-\{n\}}, v_{b,n})$.
- Output. At some point the adversary outputs its guess b'.
- Winning condition. The adversary wins the game if the following conditions hold:
  1. $b' = b$.
  2. $v_{0,i} = v_{1,i}$ for any $i \in S$.
  3. for any pair of vectors $(\boldsymbol{v}_0, \boldsymbol{v}_1)$ for which $\mathcal{A}$ asked a query to the oracle Vote it holds that: for any vector $\boldsymbol{v}$, $F(\boldsymbol{v}'_0) = F(\boldsymbol{v}'_1)$ where for $b = 0, 1$ $\boldsymbol{v}'_b$ is the vector equal to $\boldsymbol{v}$ in all indices in S and equal to $\boldsymbol{v}_b$ elsewhere.
  4. S has cardinality $< n$, $\boldsymbol{v}_0$ and $\boldsymbol{v}_1$ are vectors of n values in D and $\mathsf{id} \in \{0,1\}^\lambda$.

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

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{NIVS},\mathsf{Priv}}(1^\lambda) \stackrel{\triangle}{=} |\mathrm{Prob}[\mathsf{Priv}_{\mathcal{A}}^{n,D\Sigma,F,\mathsf{NIVS}}(1^\lambda) = 1] - 1/2|$$

Definition 2 We say that NIVS for parameters (n, D, Σ, F) is *private* if all PPT adversaries $\mathcal{A} \stackrel{\triangle}{=} (\mathcal{A}_0, \mathcal{A}_1)$ have at most negligible advantage in the above game.

Remark 1 We make some remarks on the previous definitions.

- **Perfect synchronous broadcast channel.** Our security definition implicitly assumes a synchronous broadcast channel and as such does not model e.g. malleability and copying attacks.
- **Parameterization.** A (n, D, Σ, F)-NIVS is fully specified only for the 4 parameters n, D, Σ and F, but often for simplicity we will drop the parameters and we will talk about a NIVS when it is clear from the context.
- **Supporting multiple functions.** It is possible to extend the definition of a (n, D, Σ, F)-NIVS by replacing the function F with a *set* of functions F so as to have a system that in each election can allow to evaluate the tally according to any function $f \in \mathsf{F}$. In this case, the setup algorithm has to take as additional input a finite description of the set and the other algorithms have to take as additional input a certain function $f \in \mathsf{F}$. The correctness, verifiability and privacy have to be changed accordingly. We point out that our NIVSs for 0/1 elections, for dead or alive elections and for elections by unanimity can be easily unified in a single NIVS for the set of the three corresponding functions.
- **Verifiability.** Note that the first part of the verifiability states that a ballot uniquely determines a single vote v that is compatible with any other correctly computed set of ballots. The second part guarantees that the VerifyBallot algorithm can discover whether a ballot is cast correctly. Thus, if the check is satisfies (i.e., with output OK), it means that for a given ballot Blt, any set of $n-1$ correctly computed ballots, will give consistent results.
- **Constant or polynomial number of voters.** The reader may have noticed that we leave unspecified the relation between the parameter n and the security parameter. In more cases, setting n to a constant is enough. However one could set n to be any polynomial in the security parameter.
- **Programmable RO.** Actually we will assume a definition of privacy identical to the one we formulated except that it is in the (programmable) RO model. In this case the adversary would have in addition oracle access to a function O drawn at random from the space of functions $\mathcal{O}$ that map $\{0,1\}^\lambda$ to some space Σ but possibly modified by a PPT simulator in a polynomial (in λ) number of points. We skip details of formal definitions.
  In our schemes we will assume that the adversary has access to more than one oracle, but using standard techniques it could be modified to use only one oracle but not to overburden the presentation we do not do that.
- **CRS vs public-coin model.** The public parameters can be seen as a CRS, so one can wonder whether there is difference between the public-coin model and the CRS model. The difference is that in the CRS model the party that generates the public parameters is *not* trusted (though we mention that the trust could be distributed among a set of trusted parties in a threshold way), whereas in the standard model the security should

hold even with respect to the party who generated the parameters.
The above definition of privacy only takes in account the CRS model but can be changed to the standard model by allowing the adversary to see the random coins with which the public parameters are generated. We stress that our construction of Section 3 satisfies this stronger definition assuming a variant of BDDH (see Section 2.2).

2.2 Bilinear maps

In this section we describe the bilinear setting with groups of prime order and the assumption that we will use to prove the privacy of the NIVSs presented in Sections 3 and 1.2.

Prime order bilinear groups. Prime order bilinear groups were first used in Cryptography by Boneh and Franklin [BF01], and Joux [Jou04]. We suppose the existence of an efficient group generator algorithm $\mathcal{G}$ which takes as input the security parameter λ and outputs a description $\mathcal{I} \stackrel{\triangle}{=} (p, \mathbb{G}, \mathbb{G}_T, \mathbf{e})$ of a bilinear instance of prime order, where $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups of prime order p, and $\mathbf{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a map with the following properties:

1. (Bilinearity): $\forall\ g, h \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$ it holds that $\mathbf{e}(g^a, h^b) = \mathbf{e}(g, h)^{ab}$.
2. (Non-degeneracy): $\exists\ g \in \mathbb{G}$ such that $\mathbf{e}(g, g)$ has order p in $\mathbb{G}_T$.

Bilinear Decision Diffie-Hellman Assumption. More formally, we have the following definition. First pick a random bilinear instance $\mathcal{I} \stackrel{\triangle}{=} (p, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathcal{G}(1^\lambda)$ and then pick $g \leftarrow \mathbb{G}, a, b, c, z \leftarrow \mathbb{Z}_p$, and set $D \stackrel{\triangle}{=} (\mathcal{I}, g, g^a, g^b, g^c)$, $T_0 \stackrel{\triangle}{=} \mathbf{e}(g, g)^{abc}$ and $T_1 \stackrel{\triangle}{=} \mathbf{e}(g, g)^z$. We define the advantage of any $\mathcal{A}$ in breaking the BDDH Assumption (with respect to $\mathcal{G}$) to be

$$\mathsf{Adv}^{\mathcal{A},\mathcal{G}}_{\mathsf{BDDH}}(\lambda) \stackrel{\triangle}{=} |\mathrm{Prob}[\mathcal{A}(D, T_0) = 1] - \mathrm{Prob}[\mathcal{A}(D, T_1) = 1]| \ .$$

We say that Assumption BDDH holds for generator $\mathcal{G}$ if for all non-uniform PPT algorithms $\mathcal{A}$, $\mathsf{Adv}^{\mathcal{A}(\lambda),\mathcal{G}}_{\mathsf{BDDH}}$ is a negligible function of λ.

We mention that if we wish that our NIVS of Section 3 satisfy privacy in the public-coin model, we need to assume a stronger variant of the above definition in which the adversary also sees the random coins used to generate the bilinear instance.

2.3 NIZK in the RO

Let R be an efficiently computable binary relation. For pairs $(x, w) \in R$ we call x the statement and w the witness. Let L be the language consisting of statements in R.

Definition 3 [NIZK] A non-interactive zero-knowledge proof system [BFM88,FLS90] (NIZK, in short) in the RO model [BR93,BFW15] $\mathsf{NIZK} = (\mathsf{Prove}, \mathsf{Verify})$ for a relation R consists of the following PPT algorithms with access to an oracle O randomly drawn from a space $\mathcal{O}$ of functions with domain and co-domain $\{0,1\}^\lambda$:

– $\mathsf{Prove}^{O(\cdot)}(x, w)$: takes as input a statement x and a witness w for x, and with oracle access to O produces a proof π.

– $\mathsf{Verify}^{\mathsf{O}(\cdot)}(\mathsf{x}, \pi)$: takes in input a statement x and a proof π, and with oracle access to O outputs 1 if the proof is accepted and 0 otherwise.

We call NIZK a non-interactive zero-knowledge proof system for R if it has the properties described below.

– Perfect completeness. A proof system is complete if an honest prover with a valid witness can convince an honest verifier. Formally we have that for any $(x, w) \in R$

$$\Pr[O \leftarrow \mathcal{O};\ \pi \leftarrow \mathsf{Prove}^{O(\cdot)}(x, w) : \mathsf{Verify}^{\mathsf{O}(\cdot)}(\mathsf{x}, \pi) = 1] = 1\ .$$

– Statistical soundness. A proof system is sound if it is infeasible to convince an honest verifier when the statement is false. For all (even unbounded) non-uniform adversaries A we have

$$\Pr[O \leftarrow \mathcal{O};\ \exists(x, \pi) : \mathsf{Verify}^{\mathsf{O}(\cdot)}(\mathsf{x}, \pi) = 1 \wedge \mathsf{x} \notin \mathsf{L}] = \mathsf{negl}(\lambda)\ .$$

– (Adaptive Multi-theorem) Computational zero-knowledge [BFW15]. A proof system is computational zero-knowledge[9] in the RO model if the proofs do not reveal any information about the witnesses to a bounded adversary. We say a non-interactive proof NIZK is computational zero-knowledge if there exists a PPT *stateful* simulator $\mathsf{Sim} = (\mathsf{Sim}.\mathcal{RO}, \mathsf{Sim})$ that without access to the witness can simulate proofs having in addition the capability of programming the oracle O at any point, i.e, for any x and y it is able to set $O(x) \triangleq y$. For all non-uniform PPT adversaries $\mathcal{A}$ with access to an oracle O, we have that the following quantity is negligible in λ:

$$\begin{aligned}&|\Pr[O \leftarrow \mathcal{O} : \mathcal{A}^{O(\cdot), \mathsf{Prove}_2^{O(\cdot)}(\cdot,\cdot)}(1^\lambda) = 1] - \\ &\Pr[O \leftarrow \mathcal{O} : \mathcal{A}^{\mathsf{Sim}.\mathcal{RO}^{O(\cdot)}, \mathsf{Sim}_2^{O(\cdot)}(\cdot,\cdot)}(1^\lambda) = 1]|\ ,\end{aligned}$$

where $\mathsf{Prove}_2^{O(\cdot)}(x, w) \triangleq \mathsf{Prove}^{O(\cdot)}(x, w)$ for $(x, w) \in R$, $\mathsf{Sim}_2^{O(\cdot)}(x, w) \triangleq \mathsf{Sim}^{O(\cdot)}(x)$ for $(x, w) \in R$, the latter oracles output $\perp$ for $(x, w) \notin R$ and $\mathsf{Sim}.\mathcal{RO}$ simulates the oracle O possibly modifying it at an arbitrary number of points.

NIZK in the RO for encryption of 0 or 1. Recall that Hao *et al.* used a protocol of Cramer *et al.* [CDS94] to prove that their ballot correspond to a vote of either 0 or 1. To that aim they convert the terms of their protocol into the form of ElGamal encryptions by seeing the pair (g, g^{y_i}) terms as El Gamal PKs and thus seeing the pair $g^{x_i}, g^{y_i x_i} g^{v_i}$ as an El Gamal encryption with randomness x_i, public-key g^{y_i} and plaintext v_i. The Cramer *et al.*'s sigma protocol can prove that v_i is either 0 or 1 without revealing which. Using the Fiat-Shamir's heuristic [FS87] (see also [BFW15] for discussions about adaptiveness) it can be converted in a NIZK in the RO model.

In our work we need a NIZK in the RO for a relation identical as above except that g is an element of the target group of a bilinear group. Specifically the variable g above takes the form $g \triangleq \mathbf{e}(g', \mathsf{Hash}(\mathcal{I}, s))$ where g' is an element of a bilinear group, $\mathcal{I}$ is a bilinear instance, s is some string and Hash is an hash function mapping the input to the base group.

[9] Note that our definition of zero-knowledgeness is multi-theorem and adaptive like in [BFW15].

It is straightforward to see that the protocol of Cramer *et al.* also work when g has this form. In fact the computational assumption on which the security of the sigma protocol of Cramer *et al.* depends, also holds when the underlying group is the target group of a bilinear group, and in particular when the generator of such group has the above form. This is easy to verify assuming standard assumptions on bilinear maps, but in order not to overburden the presentation we skip the details.

Precisely our relation R_{wf} is the following.

Definition 4 [Relation R_{wf}] $R_{\mathsf{wf}}(x, w) \stackrel{\triangle}{=} 1$ if $x = (\mathcal{I}, g, A, B, C)$ consists of a bilinear instance $\mathcal{I} \stackrel{\triangle}{=} (p, \mathbb{G}, \mathbb{G}_T, \mathbf{e})$ and a triple of 3 elements of $\mathbb{G}_T$ and $w = (x, y, v)$ are such that $A = g^y, B = g^x, C = g^{xy} g^v$.

3 NIVS for YES/NO elections

In this Section we present our NIVS for YES/NO elections.

Definition 5 [NIVS for YES/NO elections] Let O and O_2 be two random oracles (that in the implementation will be set to two secure hash functions, e.g., SHA3). Let $\mathcal{G}$ be a generator for a bilinear instance of prime order, let $\mathsf{NIZK} = (\mathsf{Prove}^O, \mathsf{Verify}^{\mathsf{O}})$ be a NIZK in the RO for the relation R_{wf} of Definition 4. Let $n(\lambda)$ be the number of voters, $D \stackrel{\triangle}{=} \{0,1\}$ be the domain of valid votes, $\Sigma \stackrel{\triangle}{=} [n]$ and F the sum function. Furthermore, we assume that the oracle O_2 takes as input a description of a bilinear instance $\mathcal{I} = (p, \mathbb{G}, \mathbb{G}_T, \mathbf{e})$ and maps strings from $\{0,1\}^\lambda$ to G, and that oracle O maps strings from $\{0,1\}^\lambda$ to $\{0,1\}^{p(\star)}$ for some polynomial $p(\cdot)$ as needed by NIZK.

We define a (n, D, Σ, F)-NIVS
$\mathsf{NIVS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Cast}, \mathsf{VerifyBallot}, \mathsf{EvalTally})$ in the RO model as follows.

- $\mathsf{Setup}(1^\lambda)$: on input the security parameter in unary, it outputs $\mathsf{pp} \stackrel{\triangle}{=} \mathcal{I}$ where $\mathcal{I} \stackrel{\triangle}{=} (p, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathcal{G}(1^\lambda)$.
- $\mathsf{KeyGen}(\mathsf{pp})$: on input the public parameters $\mathsf{pp} \stackrel{\triangle}{=} (g, p, \mathbb{G}, \mathbb{G}_T, \mathbf{e})$, the algorithm chooses a random $x \leftarrow \mathbb{Z}_p$ and outputs the pair $(\mathsf{Pk} \stackrel{\triangle}{=} g^x, \mathsf{Sk} \stackrel{\triangle}{=} x)$.
- $\mathsf{Cast}(\mathsf{pp}, j, \mathsf{id}, \mathsf{Sk}, (\mathsf{Pk})_{i \in [n]-\{j\}}, v)$, on input the public parameters pp, the secret-key $\mathsf{Sk} \stackrel{\triangle}{=} x$ of voter j, the identifier id of the election, the public keys $(\mathsf{Pk}_i)_{i \in [v]-\{j\}}$ of the other voters, and a vote $v \in D$, outputs a pair (Blt, π) where the *ballot* $\mathsf{Blt} \stackrel{\triangle}{=} \mathbf{e}(Y_j, O_2(\mathcal{I}, \mathsf{id}))^{\mathsf{Sk}} \cdot \mathbf{e}(g, O_2(\mathcal{I}, \mathsf{id}))^v$, where $Y_j \stackrel{\triangle}{=} \prod_{i<j} \mathsf{Pk}_i / \prod_{j>i} \mathsf{Pk}_i = g^{\sum_{i<j} x_i - \sum_{i>j} x_i}$ and π is the proof computed by $\mathsf{NIZK.Prove}^O$ with witness x_i and v of the fact that the ballot is well-formed and $v \in \{0,1\}$.
- $\mathsf{VerifyBallot}(\mathsf{pp}, \mathsf{Pk}, \mathsf{id}, \mathsf{Blt})$, on input the public parameters pp, a public-key Pk of a voter, the identifier $\mathsf{id} \in \{0,1\}^\lambda$ of the election and a ballot $\mathsf{Blt} \stackrel{\triangle}{=} (\mathsf{Blt}, \pi)$, outputs OK if $\mathsf{NIZK.Verify}^{\mathsf{O}}(\mathsf{Blt}, \pi) = 1$ or $\perp$ otherwise.
- $\mathsf{EvalTally}(\mathsf{pp}, \mathsf{Pk}_1, \ldots, \mathsf{Pk}_n, \mathsf{id}, \mathsf{Blt}_1, \ldots, \mathsf{Blt}_n)$, on input the public parameters pp, the public-keys of all voters, the identifier $\mathsf{id} \in \{0,1\}^\lambda$ of the election, and the ballots cast by all voter, computes what follows.
  It runs $\mathsf{VerifyBallot}$ on any ballot $\mathsf{Blt}_i, i \in [n]$ and if for any ballot the verification fails,

it outputs $\perp$. Otherwise it computes $R = \prod_{i\in[n]} \mathsf{Blt}_i$ and by brute force computes $r \stackrel{\triangle}{=} \mathbf{dlog}_{\mathbf{e}(g,O_2(\mathcal{I},\mathsf{id}))}R$. Finally, the algorithm outputs r.

3.1 Properties and security of the scheme

Correctness. It is straightforward to verify that the scheme satisfy the correctness as, by construction of the y_i's it follows that $\sum x_i y_i = 0$.

Verifiability. The verifiability follows from the statistical soundness of NIZK.

Privacy. We prove Theorem 1 using a standard hybrid argument. Assume by contradiction that there exist a PPT adversary $\mathcal{A}$ with non-negligible advantage in the privacy game. To that aim we define a sequence of hybrid experiments against a non-uniform PPT adversary $\mathcal{A}$ attacking the privacy game by asking at most q queries to its oracle Vote and we prove their computational indistinguishability.

- H_0. This correspond to the privacy experiment when the challenge bit is set to 0.
- H_1. This experiment is identical to H_0 except that the NIZK proofs are simulated.

  Claim 1 Indistinguishability of H_1 from H_0. The indistinguishability of the two experiments follow from the computational zero-knowledge of NIZK.

- $H_{i,j}$, for $i \in [q], j = 0, \ldots, n$. The experiment $H_{i,j}$ for $i \in [q], j = 1, \ldots, n$ is identical to H_1 except that the first $i-1$ queries are answered as if the challenge bit were $b \stackrel{\triangle}{=} 1$ (i.e., the adversary receives a set of ballots for the vector $\boldsymbol{v}_1$), and the i-th query is answered in the following way.
  Let $\boldsymbol{v}_0, \boldsymbol{v}_1$ be the two vectors for the i-th query. Please remember that the two vectors have equal Hamming weight. Through a set of intermediate vectors $\boldsymbol{v}^j$'s we want to change $\boldsymbol{v}^0$ to $\boldsymbol{v}^1$ from left to right by swapping bits. We compute a vector $\boldsymbol{v}^j$ in the following iterative way where we set $\boldsymbol{v}^0 \stackrel{\triangle}{=} \boldsymbol{v}_0$. For $j = 1, \ldots, n$ we update $\boldsymbol{v}^j$ as follows. At the beginning we set $v^j \stackrel{\triangle}{=} v^{j-1}$. If $v^{j-1}_j = v_{1,j}$ or $j = n$ then leave it unchanged, i.e, $\boldsymbol{v}^j \stackrel{\triangle}{=} \boldsymbol{v}^{j-1}$, otherwise find the next index $l_j \in [n], l_j > j$ such that $v^{j-1}_j = 1 - v^{j-1}_{l_j}$ and $v_{1,j} = 1 - v_{1,l_j}$. Such index l_j exists. In fact if $v^{j-1}_j \neq v_{1,j}$ and $j < n$ then, since the sum $\sum_i v^{j-1}_i = \sum_i v_{1,i}$ (see below), there exists at least one index l_j such that $v^{j-1}_{l_j} = 1 - v_{1,l_j}$ and the differences are opposite $v^{j-1}_{l_j} - v_{1,l_j} = -(v^{j-1}_j - v_{1,j})$. In this case we set $v^j_j \stackrel{\triangle}{=} v_{1,j}$ and $v^j_{l_j} = v_{1,l_j}$. For the same reason, for $j = n$ then $v^n_n = v_{1,n}$. Note that for any $j = 0, 1, \ldots, n$ the so formed vector $\boldsymbol{v}^j$ is such that $\sum_i v^j_i = \sum_i v_{0,i} = \sum_i v_{1,i}$, and such that $\boldsymbol{v}^j$ equals $\boldsymbol{v}_1$ in the first j positions.
  Then in experiment $H_{i,j}$ the i-th query is answered with respect to the vector $\boldsymbol{v}^j$. We set $H_{1,0} \stackrel{\triangle}{=} H_1$ and for $i = 2, \ldots, q$ we set $H_{i,0}$ to be identical to $H_{i-1,n}$,
  Note that for any $i \in [q]$ in the experiment $H_{i,n}$ the so computed vector $\boldsymbol{v}^n = \boldsymbol{v}_1$, where $\boldsymbol{v}_1$ is one of the two vectors on which the adversary queries its Vote oracle in the i-th query.
  An example of how the vector $\boldsymbol{v}^j, j = 0, \ldots, n$ is changed in the consecutive hybrid experiments is given in Figure 1.

Let $n = 5$ and $\boldsymbol{v}_0 = (10101)$ and $\boldsymbol{v}_1 = (01011)$ be the vectors asked by the adversary in query i. Then, we have the following.

- In experiment $H_{i,0}$: $\boldsymbol{v}^0 \triangleq \boldsymbol{v}_0 \triangleq (10101)$.
- In experiment $H_{i,1}$: we update $\boldsymbol{v}^1 \triangleq (01101)$ because $\boldsymbol{v}^0$ is such that $v_1^0 = 1$ and $v_{1,1} = 0$ so we search in $\boldsymbol{v}^1$ for the next index l_1 in which $\boldsymbol{v}^0$ and $\boldsymbol{v}_1$ differ (and have opposite difference) that in this case is $l_1 \triangleq 2$.
- In experiment $H_{i,2}$: we leave $\boldsymbol{v}^2 \triangleq (01101)$ unchanged because $\boldsymbol{v}^1$ is such that $v_2^1 = v_{1,2}$.
- In experiment $H_{i,3}$: we update $\boldsymbol{v}^3 \triangleq (01011)$ because $\boldsymbol{v}^2$ is such that $v_3^2 = 1$ and $v_{1,3} = 0$ so we search in $\boldsymbol{v}^3$ for the next index l_3 in which $\boldsymbol{v}^2$ and $\boldsymbol{v}_1$ differ (and have opposite difference) that in this case is $l_3 \triangleq 4$.
- In experiment $H_{i,4}$: we leave $\boldsymbol{v}^4 \triangleq (01011)$ unchanged because $\boldsymbol{v}^3$ is such that $v_4 = 1$ and $v_{1,4}^3 = 1$.
- In experiment $H_{i,5}$: we leave $\boldsymbol{v}^5 \triangleq (01011)$ unchanged because $\boldsymbol{v}^4$ is such that $v_5^4 = 1$ and $v_{1,5} = 1$.
  Note that in all experiments the vector $\boldsymbol{v}$ has the same Hamming weight.

Fig. 1. Example of how the vector $\boldsymbol{v}^j$ is iteratively updated in the hybrid experiments $H_{i,j}$'s.

Claim 2 Indistinguishability of $H_{i,j-1}$ from $H_{i,j}$, for $i \in [q], j = 1, \ldots, n$. The indistinguishability of the two experiments follow from the BDDH Assumption.

Proof. Suppose that the vector $\boldsymbol{v}^{j-1}$ used in experiment $H_{i,j-1}$ differs from $\boldsymbol{v}_1$ in position j, otherwise the experiments are identical and the proof is concluded. Let $l_j > j$ be the index such that $v_j^{j-1} = 1 - v_{l_j}^{j-1}$ and $v_{1,j} = 1 - v_{1,l_j}$. Such index exist by the assumption that the Hamming weights of $\boldsymbol{v}_0$ and $\boldsymbol{v}_1$ is equal and from the fact that for any $j = 0, \ldots, n$ the vectors $\boldsymbol{v}^j$ have same Hamming weight. Without loss of generality let us assume that $v_j^{j-1} = 1$ and $v_{l_j}^{j-1} = 0$ (the other case is symmetrical). We construct a PPT adversary $\mathcal{B}$ against the BDDH (with respect to generator of the bilinear instance $\mathcal{G}$) as follows.
$\mathcal{B}$ receives as input a bilinear instance $\mathcal{I} = (p, \mathbb{G}, \mathbb{G}_T, \mathbf{e})$ and a tuple (g, A, B, C, Z) of group elements where $A \triangleq g^a, B \triangleq g^b, C \triangleq g^c$ are random group elements of $\mathbb{G}$ and Z is either $\mathbf{e}(g,g)^{abc}$ or a random element in $\mathbb{G}_T$. $\mathcal{B}$ can use $\mathcal{I}$ to generate the public parameters pp and executes the adversary $\mathcal{A}$ on it. Then $\mathcal{A}$ outputs the set S of corrupted voters and $\mathcal{B}$ computes the PKs and SKs in the following way.
Note that S can not contain j and l_j by the constraint in the winning condition. $\mathcal{B}$ sets $\mathsf{Pk}_j = A$ and $\mathsf{Pk}_{l_j} = B$ and for any $i \neq j, l_j$ it chooses $s_i \leftarrow \mathbb{Z}_p$ and sets $\mathsf{Pk}_i = g^{s_i}$. This implicitly defines $\mathsf{Sk}_i \triangleq s_i$, for $i \neq j, l_j$, $\mathsf{Sk}_j \triangleq a$ and $\mathsf{Sk}_{l_j} \triangleq b$. Note also that $B \subseteq \{i\}_{i \neq j, l_j}$.
Therefore, $\mathcal{B}$ executes $\mathcal{A}$ with input the PKs and the SKs corresponding to set S (and it can do that as it knows the secret-keys $\mathsf{Sk}_i \triangleq s_i, i \in S$). $\mathcal{B}$ answers an oracle query id to O_2 by setting $O_2(\mathcal{I}, x) \triangleq g^{x_{\mathsf{id}}}$ for $x_{\mathsf{id}} \leftarrow \mathbb{Z}_p$ and by setting $O_2(\mathcal{I}, \mathsf{id}^\star) = C$ where $\mathsf{id}^\star$ is the identifier used by $\mathcal{A}$ in the i-th query.

With this setting, it is easy to see that $\mathcal{B}$ can simulate all queries $k \neq i$ queries using the group elements A, B, the values s_i's and x_{id}'s.
For the i-th query, $\mathcal{B}$ can set Blt_k for $k \neq j, l_j$ by using A, B, C and the values s_i's. Finally, it sets $\mathsf{Blt}_j \triangleq \mathbf{e}(A,C)^{\sum_{i<j} s_i - \sum_{i>j, i\neq l_j} s_i} \cdot Z^{-1} \cdot \mathbf{e}(g,C)^{v_j^{j-1}}$ and $\mathsf{Blt}_{l_j} \triangleq \mathbf{e}(B,C)^{\sum_{i<l_j, i\neq j} s_i - \sum_{i>j} s_i} \cdot Z \cdot \mathbf{e}(g,C)^{v_{l_j}^{j-1}}$ Note that if $Z \triangleq \mathbf{e}(g,g)^{abc}$ then Blt_j and Blt_{l_j} are distributed like in experiment $H_{i,j-1}$. For instance,
$\mathsf{Blt}_{l_j} \triangleq \mathbf{e}(B,C)^{\sum_{i<l_j, i\neq j} s_i - \sum_{i>j} s_i} \cdot Z \cdot \mathbf{e}(g,C)^{v_{l_j}^{j-1}} \triangleq$
$\mathbf{e}(g, O_2(\mathcal{I}, \mathsf{id}^\star))^{\mathsf{Sk}_{l_j}(\sum_{i<l_j, i\neq j} \mathsf{Sk}_i - \sum_{i>j} \mathsf{Sk}_i)} \cdot Z \cdot \mathbf{e}(g,C)^{v_{l_j}^{j-1}} \triangleq$
$\mathbf{e}(g, O_2(\mathcal{I}, \mathsf{id}^\star))^{\mathsf{Sk}_{l_j}(\sum_{i<l_j, i\neq j} \mathsf{Sk}_i - \sum_{i>j} \mathsf{Sk}_i)} \cdot \mathbf{e}(g,g)^{abc} \cdot \mathbf{e}(g,C)^{v_{l_j}^{j-1}} \triangleq$
$\mathbf{e}(g, O_2(\mathcal{I}, \mathsf{id}^\star))^{\mathsf{Sk}_{l_j}(\sum_{i<l_j, i\neq j} \mathsf{Sk}_i - \sum_{i>j} \mathsf{Sk}_i)} \cdot \mathbf{e}(g, O_2(\mathcal{I}, \mathsf{id}^\star))^{\mathsf{Sk}_j \mathsf{Sk}_{l_j}} \cdot \mathbf{e}(g,C)^{v_{l_j}^{j-1}} \triangleq \mathbf{e}(g^{y_j}, O_2(\mathcal{I}, \mathsf{id}^\star))^{\mathsf{Sk}_{l_j}} \cdot \mathbf{e}(g, O_2(\mathcal{I}, \mathsf{id}^\star))^{v_{l_j}^{j-1}}$.
Similarly for Blt_j. Hence, Blt_j (resp. Blt_{l_j}) is distributed correctly as output of the Cast algorithm for identifier id, set of PKs $\{\mathsf{Pk}_i\}_{i\in[n]}$, SK $\mathsf{Sk}_i \triangleq a$ (resp. $\mathsf{Sk}_{l_j} \triangleq b$) and vote v_j^{j-1} (resp. $v_{l_j}^{j-1}$).
On the other hand if Z is uniform in $\mathbb{G}_T$ then it is equal to $\mathbf{e}(g,g)^z$ for some $z \in \mathbb{Z}_p$. Setting $z \triangleq z' + \log_g C$ for $z' \triangleq z - \log_g C$ and by recalling that we assumed that $v_j^{j-1} = 1$ and $v_{l_j}^{j-1} = 0$, we see that $\mathsf{Blt}_j = \mathbf{e}(A,C)^{\sum_{i<j} s_i - \sum_{i>j, i\neq l_j} s_i} \cdot \mathbf{e}(g,g)^{-z'} \cdot \mathbf{e}(g,C)^{v_{l_j}^{j-1}}$ and $\mathsf{Blt}_{l_j} = \mathbf{e}(B,C)^{\sum_{i<l_j, i\neq j} s_i - \sum_{i>j} s_i} \cdot \mathbf{e}(g,g)^{z'} \cdot \mathbf{e}(g,C)^{v_j^{j-1}}$.
Let us call $H_i^{\mathsf{Rnd}, \boldsymbol{v}^{j-1}}$ the previous experiment simulated by $\mathcal{B}$ when Z is uniform in $\mathbb{G}_T$ and the i-th query is answered with vector $\boldsymbol{v}^{j-1}$. What we showed before implies that experiment $H_i^{\mathsf{Rnd}, \boldsymbol{v}^{j-1}}$ is distributed identically to an experiment $H_i^{\mathsf{Rnd}, \boldsymbol{v}^{j}}$ that is identical to $H_i^{\mathsf{Rnd}, \boldsymbol{v}^{j-1}}$ except that the i-th query is answered with $\boldsymbol{v}^{j}$. Moreover, if $Z \triangleq \mathbf{e}(g,g)^{abc}$ then $\mathcal{B}$ simulates perfectly experiment $H_{i,j-1}$. By BDDH, $H_{i,j-1} \approx_c H_i^{\mathsf{Rnd}, \boldsymbol{v}^{j-1}} \equiv H_i^{\mathsf{Rnd}, \boldsymbol{v}^{j}}$. Furthermore, by symmetry it is easy to see that $H_{i,j} \approx_c H_i^{\mathsf{Rnd}, \boldsymbol{v}^{j}}$, and thus we conclude that $H_{i,j-1} \approx_c H_{i,j}$ as we had to prove.

Note that the hybrid $H_{1,0}$ is by definition identical to the real experiment for bit $b \triangleq 0$ and $H_{q,n}$ is identical to the real experiment for bit $b \triangleq 1$. Thus, the indistinguishability of these two experiments implies that no PPT non-uniform adversary has non-negligible advantage in the privacy game.

4 Acknowledgments

Vincenzo Iovino is supported by the National Research Fund, Luxembourg. We thank Yu Li for useful comments and Qiang Tang to point out a generalization of our definition of dispure-freeness.

References

BF01. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, August 2001.

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, 1988.

BFW15. David Bernhard, Marc Fischlin, and Bogdan Warinschi. Adaptive proofs of knowledge in the random oracle model. 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 629–649, 2015.

Boy08. Xavier Boyen. The uber-assumption family (invited talk). In Steven D. Galbraith and Kenneth G. Paterson, editors, *PAIRING 2008: 2nd International Conference on Pairing-based Cryptography*, volume 5209 of *Lecture Notes in Computer Science*, pages 39–56. Springer, September 2008.

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.

CDS94. Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo Desmedt, editor, *Advances in Cryptology – CRYPTO'94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer, August 1994.

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

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 *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, pages 578–602, 2014.

GGHR14. Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure MPC from indistinguishability obfuscation. In *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, pages 74–94, 2014.

Gol04. Oded Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004.

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.

IZ15. Vincenzo Iovino and Karol Zebrowski. Simulation-based secure functional encryption in the random oracle model. In *Progress in Cryptology - LATINCRYPT 2015 - 4th International Conference on Cryptology and Information Security in Latin America, Guadalajara, Mexico, August 23-26, 2015, Proceedings*, pages 21–39, 2015.

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

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.

MPR06. Silvio Micali, Rafael Pass, and Alon Rosen. Input-indistinguishable computation. In *47th Annual Symposium on Foundations of Computer Science*, pages 367–378. IEEE Computer Society Press, October 2006.

Yao88. Andrew Chi-Chih Yao. Near-optimal time-space tradeoff for element distinctness. In *29th Annual Symposium on Foundations of Computer Science*, pages 91–97. IEEE Computer Society Press, October 1988.