Efficient Universally-Verifiable Electronic Voting with Everlasting Privacy

David Pointcheval

DIENS, École normale supérieure, CNRS, Inria, PSL University, Paris, France

Abstract. Universal verifiability is a must-to-have for electronic voting schemes. It is essential to ensure honest behavior of all the players during the whole process, together with the eligibility. However, it should not endanger the privacy of the individual votes, which is another major requirement. Whereas the first property prevents attacks during the voting process, privacy of the votes should hold forever, which has been called *everlasting privacy*.

A classical approach for universal verifiability is to add some proofs together with the encrypted votes, which requires publication of the latter, while eligibility needs a link between the votes and the voters: it definitely excludes long-term privacy. An alternative is the use of perfectly-hiding commitments, on which proofs are published, while ciphertexts are kept private for computing the tally.

In this paper, we show how recent linearly-homomorphic signatures can be exploited for all the proofs, leading to very efficient procedures towards universal verifiability with both strong receipt-freeness and everlasting privacy. Privacy will indeed be unconditional, after the publication of the results and the proofs, whereas the soundness of the proofs holds in the algebraic group model and the random oracle model.

1 Introduction

Electronic Voting. The first major requirement for a voting system is the privacy of the ballot. This is essential to guarantee the freedom of expression by the voters. On the other hand, it is also wildly admitted that an electronic voting system should additionally satisfy the following properties, for an end-to-end verifiability that the result really reflects the votes, without having to trust any system or any authority: individual verifiability, where each voter can check that her ballot was tallied; universal verifiability, where anybody can check that the result corresponds to the public board, and tallied ballots come from eligible voters (eligibility). Some of these properties can come in several steps:

- Individual verifiability: the more users verify, the better it is, as it can impact the reputation of the system. They want to check *Cast-as-Intended* property: the voter is sure that the actual vote corresponds to her intended choice. A classical approach is the so-called Benaloh Challenge, and variants, which is an audit-or-cast technique, in the same vein as the cut-an-choose technique. Another possibility is an open-source code on the voter-side, that can be publicly audited and easily checked at execution time; and *Recorded-as-Cast* property: the voter is sure that the vote recorded in the ballot-box corresponds to the actual vote. A simple way, in theory, is the publication of the ballot-box, that allows anybody to control the presence of her ballot;
- Universal verifiability: once everybody is convinced by data on the public board (possibly the ballot-box itself), thanks to the above individual verifiability, they check *Tallied-as-Recorded* property: the result corresponds to the data on the public board. Everybody should also be able to check eligibility of the voters.

Two major approaches exist for counting the tally from encrypted votes: either one applies a mixing-network (mixnet), which permutes and randomizes the encrypted ballots, before decryption of all the individual ballots to perform the counting in the clear, as one does with paper-based voting systems when one opens the envelops after having mixed them to remove any link with the voters; or one uses homomorphic encryption that allows to aggregate the encrypted ballots to get the encrypted tally, that is the unique value eventually decrypted. However, both approaches require publication of the ballot-box, with all the encrypted votes, for being universally verifiable. And eligibility verifiability requires a link between the ballots and the voters. This is a risk if the encryption mechanism gets later broken, or if the decryption key is leaked.

Everlasting Security. Publishing all the encrypted votes is indeed a huge threat, as any public-key encryption scheme will possibly get broken in the future, either because of new algorithms or new kinds of computers, or just because of the key-sizes that will become too small for the new technologies.

In a recent survey [HMMP23] on everlasting electronic voting systems, they defined two families of protocols: B-ANON, which means the published information is anonymous; and B-ID, where the published information is related to the voters. In the former class, everlasting privacy reduces to publishing encrypted ballots anonymously, which has a strong impact on the individual verifiability and the eligibility property; while in the latter class, public information is authenticated by the voters. They thus argue that B-ID is superior to B-ANON. But this is more difficult to achieve. The best candidate is [CPP13], with *Commitment Consistent Encryption* (CCE), where a perfectly hiding commitment is associated to each ciphertext. It is illustrated with ElGamal encryption [ElG85] and Pedersen commitment [Ped92]. This commitment is perfectly hiding: even a powerful adversary cannot recover the committed value. But it requires some additional proofs that might make the approach not homomorphic anymore, and not efficient for complex ballots. They extended the primitive to CCE with Validity Augmentation (named CCVAE). A first solution then requires Paillier encryption [Pai99], which makes it inefficient. A second solution combines ElGamal encryption with the TC2 perfectly hiding commitment [AHO10], which requires group elements in both groups in a type III pairing-friendly setting, with security proof under the SXDH assumption.

The survey [HMMP23] concludes that this approach with CCVAE is among the best, but may not be appropriate for complex ballots (for the homomorphic version) and does not provide (strong) receipt-freeness. It is thus essentially applicable with mixnets. But the mixing operation is quite long to generate, and all the individual ballots must be decrypted, which can be prohibitive for large-scale elections. Homomorphic encrypted tally is definitely the most appropriate solution, when the tally just consists in counting the number of votes for each candidates, as it allows a fast publication of the results, if verifications and aggregations are performed on-the-fly.

Strong Receipt-Freeness. As explained by Cortier and Smyth [CS11], privacy is more complex than it appears, in particular when legitimate voters are ready to change their votes for money or to break privacy of another vote. More advanced protections have to be considered: one must avoid replay attacks, where voters can amplify Alice's vote to learn her vote; and receipts that allow a voter to convince a buyer of the content of her vote. Strong receipt-freeness prevents both attacks [CCFG16,CFL19].

A usual approach for excluding receipts is to randomize the ballots so that the voter does not know anymore how to open it, but this should keep the validity proofs correct. Hence the need of randomizable encryption/commitment with randomizable proofs of validity: Schnorr-like proofs cannot be used by the voters. We will use linearly-homomorphic signatures with randomizable tags, as introduced in [HPP20,Poi23], as they offer short and efficient homomorphic quasi-adaptive NIZKs, which is well-suited for e-voting, with constant-time generation of the proof, and the randomizability of the proof.

Contributions. We follow the path of the CCE/CCVAE primitives for homomorphic encrypted tally: the ballot sent by the voter will consist of an ElGamal Ciphertext of the vector $\vec{m}$ of the votes (e.g. a vector of 0/1 to cast multiple yes/no ticks in boxes), a Pedersen Commitment of that vector $\vec{m}$, an encryption of the opening, a proof of consistency (the encrypted message and the committed message are the same), and a proof of validity (the committed message is of correct format).

Thanks to the homomorphic property of ElGamal encryption, the usual homomorphic encrypted tally can be computed, and decrypted in a distributed way. The Pedersen commitment being also additively homomorphic, one can easily get a Pedersen commitment of the tally, and the corresponding opening value exploits the homomorphic property of the encryption of each individual opening values. We essentially extend [CPP13] to complex ballots. And using linearly homomorphic signatures with randomizable tags [HPP20,Poi23], we show that proofs of both consistency and validity can be efficiently generated, still being compatible with strong receipt-freeness. But contrarily to [Poi23] that suggested the use of square Diffie-Hellman tags, we will use the more efficient FHS signature [FHS19], as in [HPP20]: thanks to their perfect randomizability, we will get perfect privacy.

While privacy will be unconditional, soundness of the various verification steps will rely on Discrete-Logarithm-like assumptions in the Algebraic Group Model (AGM) [FKL18] and the Random Oracle Model (ROM) [BR93]. This is the cost to pay with our approach, compared to the original CCE/CCVAE paper [CPP13]. But our goal is to obtain a practical solution, with efficient ballot generation for the voter. One should note that our computational assumptions (for the soundness only) are similar to the one required in cryptocurrencies, and namely for privacy with practical zk-SNARKs [BCCT13,GGPR13], such as Groth16 [Gro16], which are even proven in the Generic Group Model only. But contrarily to most of them, the CRS can be efficiently performed in a distributed way in our case.

Organization. Whereas FHS signatures [FHS19] have already been proven unforgeable in the Generic Group Model, as signatures on equivalent classes [BF20], we prove them unforgeable, as linearly-homomorphic signatures, in the Algebraic Group Model, under Discrete-Logarithm-like assumptions (see Section 3). While only under selective-message attacks, this unforgeability is enough for getting the soundness of the proofs in our electronic voting scheme (see Section 4). Eventually, we provide some benchmarks to illustrate concrete efficiency of our approach (see Section 5).

2 Preliminaries

2.1 Computational Assumptions

Our analysis will be performed in the Algebraic Group Model (AGM) [FKL18], where any algorithm is assumed to be algebraic: any output group element comes together with a linear combination of the input group elements. But we still require specific computational assumptions: first, in a group $\mathbb{G}$ of prime order p, spanned by a public generator G, we have

Discrete Logarithm (DL) From $U = xG$, for $x \xleftarrow{\$} \mathbb{Z}_p$, it is hard to compute x.

Computational Diffie-Hellman (CDH) **Assumption.** Given a generator G, and $(U = x \cdot G, V = y \cdot G)$, for $x, y \xleftarrow{\$} \mathbb{Z}_p$, it is hard to compute $xy \cdot G$;

Decisional Diffie-Hellman (DDH) For $U \xleftarrow{\$} \mathbb{G}$, $x, y \xleftarrow{\$} \mathbb{Z}_p$, the distributions $\mathcal{D}_{\mathsf{dh}} = \{(G, xG, U, xU)\}$ and $\mathbb{G}^4_\$ = \{(G, xG, U, yU)\}$ are hard to distinguish;

In a type III pairing-friendly setting $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, p, G, \hat{G}, e)$, where $\mathbb{G}$ and $\hat{\mathbb{G}}$ are groups of prime order p, spanned by public generators, respectively G and $\hat{G}$, and e is a bilinear map from $\mathbb{G} \times \hat{\mathbb{G}}$ into the target group $\mathbb{G}_T$, we have additional assumptions:

Symmetric eXternal Discrete Logarithm (SXDL) From both $U = xG$ and $\hat{U} = x\hat{G}$, for $x \xleftarrow{\$} \mathbb{Z}_p$, it is hard to compute x;

Symmetric eXternal Square Discrete Logarithm (SXSDL) From $U = xG$, $V = x^2G$, and $\hat{U} = x\hat{G}$, for $x \xleftarrow{\$} \mathbb{Z}_p$, it is hard to compute x;

Symmetric eXternal Diffie-Hellman (SXDH) The DDH assumption holds in both groups $\mathbb{G}$ and $\hat{\mathbb{G}}$;

2.2 Linearly-Homomorphic Signatures

The notion of homomorphic signatures dates back to [JMSW02], with notions in [ABC$^+$12], but the linearly-homomorphic signatures, that allow to sign vector subspaces, were introduced in [BFKW09], with several follow-up by Boneh and Freeman [BF11b,BF11a] and formal security definitions in [Fre12]. In another direction, Abe *et al.* [AFG$^+$10] proposed the notion of structure-preserving signature, where keys, messages and signatures all belong in the same group. Later Libert *et al.* [LPJY13] combined both notions and proposed a linearly-homomorphic signature scheme, that is furthermore structure-preserving. More recently, those linearly-homomorphic signatures have been applied in various protocols [HPP20,HP22,Poi23], and we follow this path here.

Definition. We start with the formal definition of linearly-homomorphic signature scheme with tags, and the security requirement, the so-called *unforgeability*, where the adversary should not be able to generate signatures of messages that are not in linear subspaces, identified by tags. We also deal with randomizable tags, that will be the core of our privacy properties.

Definition 1 (Linearly-Homomorphic Signature Scheme with Tags (LH-Sign-Tag)). *A linearly-homomorphic signature scheme with tags, for messages in $\mathbb{G}^n$ and a set $\mathcal{T}$ of tags, consists of the algorithms* ($\mathsf{Setup}, \mathsf{NewTag}, \mathsf{VerifTag}, \mathsf{RandTag}, \mathsf{Keygen}, \mathsf{Sign}, \mathsf{DerivSign}, \mathsf{Verif}$)*:*

$\mathsf{Setup}(1^\kappa)$**:** *Given a security parameter κ, it outputs the global parameter* param*, which includes the tag space $\mathcal{T}$;*

$\mathsf{Keygen}(\mathsf{param}, n)$**:** *Given the parameters* param *and an integer n, it outputs a signing-verification key pair* $(\mathsf{sk}, \mathsf{vk})$*. We will assume that* vk *implicitly contains* param *and* sk *implicitly contains* vk*;*

$\mathsf{NewTag}(\mathsf{param})$**:** *Given the parameters* param*, it outputs a verifiable tag* Tag *and, possibly, its associated secret tag τ;*

$\mathsf{VerifTag}(\mathsf{param}, \mathsf{Tag})$**:** *Given the parameters* param *and a tag* Tag*, it outputs 1 if the tag is valid and 0 otherwise;*

$\mathsf{Sign}(\mathsf{sk}, \mathsf{Tag}[, \tau], \vec{M})$**:** *Given a signing key* sk*, a verifiable tag* Tag*, possibly the associated secret tag τ, and a vector-message $\vec{M} = (M_i)_i \in \mathbb{G}^n$, it outputs the signature σ under the tag* Tag*;*

$\mathsf{RandTag}(\mathsf{vk}, \mathsf{Tag}, \vec{M}, \sigma)$**:** *Given a verification key* vk*, any verifable tag* Tag *and a signature σ on a vector-message $\vec{M}$, it outputs a new verifiable tag* Tag' *and a new signature σ'.*

$\mathsf{DerivSign}(\mathsf{vk}, \mathsf{Tag}, (\omega_j, \vec{M}_j, \sigma_j)_{j=1}^{\ell})$**:** *Given a verification key* vk*, a verifiable tag* Tag *and ℓ tuples of weights $\omega_j \in \mathbb{Z}_p$ and signed messages $\vec{M}_j$ in σ_j, it outputs a signature σ on the vector $\vec{M} = \sum_{j=1}^{\ell} \omega_j \cdot \vec{M}_j$ under the tag* Tag*;*

$\mathsf{Verif}(\mathsf{vk}, \mathsf{Tag}, \vec{M}, \sigma)$**:** *Given a verification key* vk*, a verifiable tag* Tag*, a vector-message $\vec{M}$ and a signature σ, it outputs* 1 *if both the tag* Tag *is valid and σ is also valid relative to* vk *and* Tag*, and* 0 *otherwise.*

The $\mathsf{DerivSign}$ algorithm allows linear combinations of signatures under the same tag: for any key-pair $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Keygen}(\mathsf{param}, n)$, if $\mathsf{Verif}(\mathsf{vk}, \mathsf{Tag}, \vec{M}_j, \sigma_j) = 1$ for any tag Tag and message-signature pairs $(\vec{M}_j, \sigma_j)$ for $j = 1, \ldots, \ell$, and σ is defined as $\mathsf{DerivSign}(\mathsf{vk}, \mathsf{Tag}, \{\omega_j, \vec{M}_j, \sigma_i\}_{j=1}^{\ell})$ from some scalars ω_j, then we should get $\mathsf{Verif}(\mathsf{vk}, \mathsf{Tag}, \sum_{j=1}^{\ell} \omega_j \cdot \vec{M}_j, \sigma) = 1$.

Unforgeability. However, other combinations should not be possible. This is the unforgeability notion for linearly-homomorphic signatures, we formalize against selective message attacks using random tags:

Definition 2 (Unforgeability for LH-Sign-Tag under Selective Message Attacks). *For a LH-Sign-Tag scheme, for any probabilistic polynomial time adversary $\mathcal{A}$ that*

1. *outputs K lists of messages $(\vec{M}_{k,j})_{k,j}$, for $j = 1, \ldots, J_k$ and $k = 1, \ldots, K$;*
2. *receives a verification key* vk*, random verifiable tags* $(\mathsf{Tag}_k)_k$ *and signatures $(\sigma_{k,j})_{k,j}$ of $(\vec{M}_{k,j})_{k,j}$ under* $(\mathsf{Tag}_k)_k$*;*
3. *outputs a new valid tuple* $(\mathsf{Tag}, \vec{M}, \sigma)$

then, with overwhelming probability, there exist an index $k \in \{1, \ldots, K\}$ and coefficients $(\omega_j)_{j=1,\ldots,J_k}$ such that $\vec{M} = \sum_{j=1}^{J_k} \omega_j \cdot \vec{M}_{k,j}$.

This unforgeability notion essentially says that any derived signature is for a message $\vec{M} \in \langle (\vec{M}_{k,i})_i \rangle$ for some k. The K lists of messages define K vector subspaces and $\vec{M}$ must lie in one of them. Contrarily to [LPJY13], we do not expect $\mathsf{Tag} = \mathsf{Tag}_k$, as we will allow randomizability of the tags. Unfortunately, this is not necessarily a falsifiable definition [BF24]: in many cases, this might be hard to decide whether the output of the adversary is a valid forgery or not. But we might have a stronger unforgeability notion, that is falsifiable, when it requires a way to verify the linear relation:

Definition 3 (Extractable Unforgeability for LH-Sign-Tag under Selective Message Attacks). *As in Definition 2, for the output of a new valid tuple* $(\mathsf{Tag}, \vec{M}, \sigma)$ *from a probabilistic polynomial time adversary $\mathcal{A}$, there exists an extractor $\mathcal{E}_{\mathcal{A}}$ that outputs an index $k \in \{1, \ldots, K\}$ and coefficients $(\omega_j)_{j=1,\ldots,J_k}$ such that $\vec{M} = \sum_{j=1}^{J_k} \omega_j \cdot \vec{M}_{k,j}$, with overwhelming probability.*

LH-Sign-RTag. For some privacy reasons, we will expect an additional property on the tags, we call randomizability [HPP20], hence the linearly-homomorphic signatures with randomizable tags (LH-Sign-RTag). The correctness requires that if $\mathsf{Verif}(\mathsf{vk}, \mathsf{Tag}, \vec{M}, \sigma) = 1$, then $\mathsf{Verif}(\mathsf{vk}, \mathsf{Tag}', \vec{M}, \sigma') = 1$, for the randomized tag Tag' and signature σ'. The security notion is defined below.

Definition 4 (Tag Randomizability). *An LH-Sign-RTag has computational (resp. statistical, or perfect) tag randomizability if, given a tuple $(\mathsf{vk}, \mathsf{Tag}, \vec{M}, \sigma)$ that is valid, the distance between the distributions of (Tag', σ') that comes either from the randomization (*i.e.*, $(\mathsf{vk}, \mathsf{Tag}', \vec{M}, \sigma') \leftarrow \mathsf{RandTag}(\mathsf{vk}, \mathsf{Tag}, \vec{M}, \sigma)$), or as a fresh tag (*i.e.*, $(\mathsf{Tag}', \tau') \leftarrow \mathsf{NewTag}(\mathsf{param})$, $\sigma' \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{Tag}', \tau', \vec{M})$) is computationally negligible (resp. statistically negligible, or zero).*

This property provides unlinkability. With perfect randomizability of the tags (the two above distributions are equal) we get unconditional unlinkability.

3 FHS as a Secure LH-Sign-RTag

In the following, we will use the FHS signature [FHS19,BF20], as in [HPP20]. Unforgeability was already proven in the Generic Group Model (GGM) in the original paper [BF20], for the equivalent-class notion only. In this paper, we prove the Extractable Unforgeability (Definition 3 for LH-Sign-Tag) under the SXSDL assumption, in the Algebraic Group Model (AGM) [FKL18]. This FHS signature is preferable to the square-Diffie-Hellman scheme used in [Poi23], as the latter only provides computational tag randomizability, and perfect tag randomizability will be essential for our privacy goals.

3.1 Linearly-Homomorphic Signature with Randomizable Tags

In a type III pairing-friendly setting, we define the LH-Sign-RTag:

$\mathsf{Setup}(1^\kappa)$**:** Given a security parameter κ, it outputs param, that contains a pairing-friendly setting $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, p, P, \hat{P}, e)$;

$\mathsf{Keygen}(\mathsf{param}, n)$**:** Given param and an integer n, it generates $\mathsf{sk} = \vec{s} \xleftarrow{\$} \mathbb{Z}_p^n$, sets $\mathsf{vk} = \vec{s} \cdot \hat{P} = (\hat{P}_i = s_i \cdot \hat{P})_{i=1}^n \in \hat{\mathbb{G}}^n$, and outputs the key pair $(\mathsf{sk}, \mathsf{vk})$;

$\mathsf{NewTag}(\mathsf{param})$**:** Generates a verifiable tag $\mathsf{Tag} = (Q = 1/\tau \cdot P, \hat{Q} = 1/\tau \cdot \hat{P})$, for a random scalar $\tau \xleftarrow{\$} \mathbb{Z}_p$, the secret tag;

$\mathsf{VerifTag}(\mathsf{Tag})$**:** Parse $\mathsf{Tag} = (Q, \hat{Q})$ and checks whether $e(P, \hat{Q}) = e(Q, \hat{P})$;

$\mathsf{Sign}(\mathsf{sk}, \mathsf{Tag}, \tau, \vec{M})$**:** Given a signing key sk, a secret tag τ, and a vector-message $\vec{M} = (M_i)_i \in \mathbb{G}^n$, it outputs the signature $\sigma = \tau \cdot \sum s_i \cdot M_i = \tau \cdot {}^t\vec{s} \cdot \vec{M} \in \mathbb{G}$;

$\mathsf{RandTag}(\mathsf{vk}, \mathsf{Tag}, \vec{M}, \sigma)$**:** Given a verification key vk, a verifiable tag $\mathsf{Tag} = (Q, \hat{Q})$ and a signature σ on a vector $\vec{M}$, it chooses a random $\gamma \xleftarrow{\$} \mathbb{Z}_p$ and outputs both $\mathsf{Tag}' = (Q' = 1/\gamma \cdot Q, \hat{Q}' = 1/\gamma' \cdot \hat{Q})$ and $\sigma' = \gamma \cdot \sigma$;

$\mathsf{DerivSign}(\mathsf{vk}, \mathsf{Tag}, (w_j, \vec{M}_j, \sigma_j)_{j=1}^\ell)$**:** Given a verification key vk and ℓ tuples of weights $w_j \in \mathbb{Z}_p$ and signed messages $\vec{M}_j$ in σ_j, under the same tag Tag, it outputs the signature $\sigma = \sum_{j=1}^\ell w_j \cdot \sigma_j$, on the vector $\vec{M} = \sum_{j=1}^\ell w_j \cdot \vec{M}_j$, valid under the same tag Tag;

$\mathsf{Verif}(\mathsf{vk}, \mathsf{Tag}, \vec{M}, \sigma)$**:** Given a verification key vk, a verifiable tag $\mathsf{Tag} = (Q, \hat{Q})$, a vector-message $\vec{M}$ and a signature σ, it outputs 1 if the tag Tag is valid (*i.e.*, $e(P, \hat{Q}) = e(Q, \hat{P})$) and $e(\sigma, \hat{Q}) = \prod e(M_i, \hat{P}_i)$, and 0 otherwise.

3.2 Security Properties

One contribution of this work is a proof of unforgeability under selective message attacks in the AGM relative to the SXSDL. But in order to get a falsifiable result, we prove Extractable Unforgeability in the AGM:

Theorem 5 (Extractable Unforgeability). *Breaking extractable unforgeability of* $(\mathsf{Setup}, \mathsf{NewTag}, \mathsf{VerifTag}, \mathsf{RandTag}, \mathsf{Keygen}, \mathsf{Sign}, \mathsf{DerivSign}, \mathsf{Verif})$*, under selective message attacks is equivalent to breaking the* SXSDL *assumption in the AGM.*

The detailed proof can be found in the Appendix A, but it basically works in two steps: first, one shows that the output tag-signature only involves one tag; then one shows the output message is an explicit linear combination of the messages already signed under this tag. The reduction relies on both the SXSDL and the SXDL assumptions, but the former is the strongest, as breaking SXDL allows to break SXSDL. This proves extractable unforgeability under selective message attacks, when keys and tags are honestly generated.

Furthermore, this is clear that the tag randomizability is perfect, as it generates a truly random new pair, which will provide our everlasting privacy:

Theorem 6 (Tag Randomizability). *Tag randomizability of* $(\mathsf{Setup}, \mathsf{NewTag}, \mathsf{VerifTag}, \mathsf{RandTag}, \mathsf{Keygen}, \mathsf{Sign}, \mathsf{DerivSign}, \mathsf{Verif})$ *is perfect.*

When the unique tag $(1_{\mathbb{G}}, 1_{\hat{\mathbb{G}}})$ is used, a unique vector subspace is defined by the initial vectors. We can ignore the tag. This is thus a Linearly-Homomorphic Signature scheme ($\mathsf{LH\text{-}Sign}$) without tags.

4 Our Global Voting System

4.1 Format of the Ballot

Following the approach from [CPP13], we will consider a Secret Ballot-Box, denoted SBB, and a Public Bulletin-Board, denoted PBB. The former will contain information available to the server only, to be able to proceed to the tally; while the latter will contain information to allow universal verifiability. No integrity will be needed in the SBB, this is the responsibility of the server, only the PBB must be correct: individual verifiability will allow to get confidence in the PBB.

Hence, the ballot sent to the server will contain two parts, one for the SBB and one for the PBB. We first model the vote as a vector $\vec{m}$, expected in $\{0,1\}^n$ to express checked (1) or unchecked (0) boxes. We then combine an ElGamal [ElG85] ciphertext $(C_0, \vec{C})$ and a Pedersen [Ped92] commitment D of $\vec{m}$, with a verification tag V, for two random scalars $r, s \stackrel{\$}{\leftarrow} \mathbb{Z}_p$, in a group $\mathbb{G}$ of prime order p, with a generator P:

$$C_0 = r \cdot P \qquad \vec{C} = \vec{m} \cdot P + r \cdot \vec{Z} \qquad D = {}^t\vec{m} \cdot \vec{Q} + s \cdot Q_0 \qquad V = s \cdot Y$$

where $\vec{Z} = \vec{z} \cdot P \in \mathbb{G}^n$ is the public encryption key (and $\vec{z}$ the private decryption key, that can be generated in a distributed way among the electoral board), $Y = y \cdot Q_0 \in \mathbb{G}$ is the public verification key (and y the private verification key, that can be generated in a distributed way too), and Q_0, $\vec{Q} = (Q_i)_i$ are independent random generators of $\mathbb{G}$. The scalars r, s are called the encryption randomness and the commitment randomness, respectively. The voter eventually provides two additional proofs:

– a proof π of *consistency* of $\mathbf{C} = (C_0, \vec{C}, D, V)$, that ensures the **existence** of the witnesses $(\vec{m}, r, s)$;
– a proof $\mathit{\Pi}$ of *validity* of D, that ensures the **knowledge** of the witness $(\vec{m}, s)$, with possibly more restrictions about $\vec{m}$. We expect $\mathit{\Pi}$ to be a perfectly zero-knowledge proof of knowledge;

In order to be sure that the ciphertext $(C_0, \vec{C})$ contains a message that is committed in D, where V is a kind of opening information, the server must verify the proof π. This is its own responsibility, as in the bad case, the following verifiability will fail. For the universal verifiability explained below, when we target homomorphic tally, we additionally need the guarantee the vote $\vec{m}$ (which knowledge is proven in $\mathit{\Pi}$) is of the appropriate format (e.g., with 0 or 1, only, or more generally $\vec{m} \in \mathcal{S}$), which will be proven with $\mathit{\Pi}$ too.

After the verification of π, this ballot will be split in two parts: the private part $(C_0, \vec{C}, V)$ and the public part $(D, \mathit{\Pi})$. The former being stored (or even aggregated on-the-fly) in the secret ballot-box SBB and the latter on the public bulletin-board PBB.

4.2 Tally Computation and Verification

While the secret ballot-box receives the votes, in an encrypted way in $(C_0, \vec{C})$, the public bulletin-board does not contain any information: because of the randomness of s, $\vec{m}$ is perfectly hidden in the Pedersen commitment D. With the perfect zero-knowledge property of $\mathit{\Pi}$, we have the everlasting privacy with only the public bulletin-board. All the process will exploit the homomorphic properties of both the ElGamal ciphertext and the Pedersen commitment on all the voters $\mathcal{V}$:

– everybody can aggregate the commitments in PBB, that is correct thanks to the individual verifiability performed by all the voters, and compute

$$E = \sum_{\mathcal{V}} D_{\mathcal{V}} = \sum_{\mathcal{V}} {}^t\vec{m}_{\mathcal{V}} \cdot \vec{Q} + \sum_{\mathcal{V}} s_{\mathcal{V}} \cdot Q_0 = {}^t\left(\sum_{\mathcal{V}} \vec{m}_{\mathcal{V}}\right) \cdot \vec{Q} + \left(\sum_{\mathcal{V}} s_{\mathcal{V}}\right) \cdot Q_0$$

– the server can aggregate, on its own and possibly on-the-fly to reduce storage, the values in the SBB, and publish when the election closes:

$$\begin{aligned} F_0 &= \sum_{\mathcal{V}} C_{\mathcal{V},0} & \vec{F} &= \sum_{\mathcal{V}} \vec{C}_{\mathcal{V}} & W &= \sum_{\mathcal{V}} V_{\mathcal{V}} \\ &= \left(\sum_{\mathcal{V}} r_{\mathcal{V}}\right) \cdot P & &= \left(\sum_{\mathcal{V}} \vec{m}_{\mathcal{V}}\right) \cdot P + \left(\sum_{\mathcal{V}} r_{\mathcal{V}}\right) \cdot \vec{Z} & &= \left(\sum_{\mathcal{V}} s_{\mathcal{V}}\right) \cdot Y \end{aligned}$$

The aggregation E can be trusted: because of the individual verifiability in the PBB, where every voter $\mathcal{V}$ can check her own value $D_{\mathcal{V}}$, all the $D_{\mathcal{V}}$'s are correct, and so E is correct too. However, the other aggregations come from the secret ballot-box and the server, that are not trusted. They have to be verified later relative to E only.

Thanks to the decryption key $\vec{z}$, the electoral board can compute $\vec{m}_T \cdot P = (\sum_{\mathcal{V}} \vec{m}_{\mathcal{V}}) \cdot P = \vec{F} - \vec{z} \cdot F_0$, and then the tally $\vec{m}_T$, as this should be only small discrete logarithms, which can be made public.

This vector $\vec{m}_T$, if correct, is indeed the result of the election: $\vec{m}_T = \sum_{\mathcal{V}} \vec{m}_{\mathcal{V}}$. But it needs to be publicly verifiable to be trusted by everybody: From this alleged value $\vec{m}_T$, anybody can recover $S = (\sum_{\mathcal{V}} s_{\mathcal{V}}) \cdot Q_0 = E - {}^t\vec{m}_T \cdot \vec{Q}$. Then, with a Schnorr-like zero-knowledge proof π_R, the electoral board can show there exists $y \in \mathbb{Z}_p$ so that both $Y = y \cdot Q_0$ and $W = y \cdot S$. To this aim, the prover chooses a random $\rho \in \mathbb{Z}_p$, and sends $R = \rho \cdot Q_0$ and $T = \rho \cdot S$. From the random challenge $e = \mathcal{H}(Q_0, S, Y, W, R, T) \in \mathbb{Z}_p$, it computes $y' = \rho - e \cdot y \bmod p$ which should satisfy both $R = y' \cdot Q_0 + e \cdot Y$ and $T = y' \cdot S + e \cdot W$. The proof thus consists of $\pi_R = (e, y')$ such that $e = \mathcal{H}(Q_0, S, Y, W, y' \cdot Q_0 + e \cdot Y, y' \cdot S + e \cdot W)$.

Note that this proof of Diffie-Hellman tuple for (Q_0, S, Y, W) is statistically sound: even a powerful adversary has a negligible chance to forge a valid proof for a wrong tuple, after a polynomial number of queries to the random oracle $\mathcal{H}$. The SBB can then be deleted as well as the secret keys y and $\vec{z}$.

4.3 Security Properties

During the global process, the only public information is the public bulletin-board PBB, which helps to compute the trusted aggregation E, the result $\vec{m}_T$ and the opening value S, that is proven valid with the above π_R.

Privacy. As already noted, the only public information in PBB is $(D_{\mathcal{V}}, \Pi_{\mathcal{V}})$ for each voter $\mathcal{V}$, a perfectly hiding commitment, that does not contain any information about the vote, and an additional proof that is perfectly zero-knowledge, we thus have perfect privacy, forever, from only public information: note that the server can aggregate on-the-fly the secret information $(C_{\mathcal{V},0}, \vec{C}_{\mathcal{V}}, V_{\mathcal{V}})$ into $(F_0, \vec{F}, W)$, which means that no sensitive information is kept. Only $(D_{\mathcal{V}}, \Pi_{\mathcal{V}})$ is individually kept and published in the PBB.

Theorem 7 (Everlasting Privacy). *Given the public information PBB, $\vec{m}_T$, E, S and π_R, if the proofs $(\Pi_{\mathcal{V}})_{\mathcal{V}}$ and π_R are perfectly zero-knowledge, all the individual votes are perfectly hidden, conditioned to the tally $\vec{m}_T$.*

Proof. In the honest-but-curious setting, the distribution of the public view is $\mathcal{D}_0 = \{(\mathcal{V}, D_{\mathcal{V}} = {}^t\vec{m}_{\mathcal{V}} \cdot \vec{Q} + s_{\mathcal{V}} \cdot Q_0, \Pi_{\mathcal{V}})_{\mathcal{V}}, \vec{m}_T = \sum \vec{m}_{\mathcal{V}}, E = \sum_{\mathcal{V}} D_{\mathcal{V}}, S = E - {}^t\vec{m}_T \cdot \vec{Q}, Y = y \cdot Q_0, W = y \cdot S, \pi_R\}$, for the voter's choices $\vec{m}_{\mathcal{V}}$, and random coins $s_{\mathcal{V}} \xleftarrow{\$} \mathbb{Z}_p$, with the proofs honestly generated using the random coins as witnesses. This distribution can be first replaced by a distribution $\mathcal{D}_1$ that involves the simulators $\mathcal{S}$ and $\mathcal{S}_R$ of the perfectly zero-knowledge proofs: $\mathcal{D}_1 = \{(\mathcal{V}, D_{\mathcal{V}} = {}^t\vec{m}_{\mathcal{V}} \cdot \vec{Q} + s_{\mathcal{V}} \cdot Q_0, \Pi_{\mathcal{V}} = \mathcal{S}(D_{\mathcal{V}}))_{\mathcal{V}}, \vec{m}_T, E = {}^t\vec{m}_T \cdot \vec{Q} + (\sum s_{\mathcal{V}}) \cdot Q_0, S = E - {}^t\vec{m}_T \cdot \vec{Q}, Y = y \cdot Q_0, W = y \cdot S, \pi_R = \mathcal{S}_R(Q_0, S, Y, W)\}$ where $\vec{m}_T = \sum \vec{m}_{\mathcal{V}}$, which is equal to $\mathcal{D}_2 = \{(\mathcal{V}, D_\mathcal{V} = {}^t\vec{m}_\mathcal{V} \cdot \vec{Q} + s_\mathcal{V} \cdot Q_0, \mathit{\Pi}_\mathcal{V} = \mathcal{S}(D_\mathcal{V}))_\mathcal{V}, \vec{m}_T, E = {}^t\vec{m}_T \cdot \vec{Q} + s \cdot Q_0, S = s \cdot Q_0, Y = y \cdot Q_0, W = y \cdot S, \pi_R = \mathcal{S}_R(Q_0, S, Y, W)\}$ where $\vec{m}_T = \sum \vec{m}_\mathcal{V}$ and $s = \sum s_\mathcal{V}$. Now, we choose $s \stackrel{\$}{\leftarrow} \mathbb{Z}_p$ and a particular voter $\mathcal{V}^*$: for all $\mathcal{V} \neq \mathcal{V}^*$, $D_\mathcal{V} \stackrel{\$}{\leftarrow} \mathbb{G}$, and $D_{\mathcal{V}^*} = {}^t\vec{m}_T \cdot \vec{Q} + s \cdot Q_0 - \sum_{\mathcal{V} \neq \mathcal{V}^*} D_\mathcal{V}$. Then, the new distribution $\mathcal{D}_3 = \{(\mathcal{V}, D_\mathcal{V}, \mathit{\Pi}_\mathcal{V} = \mathcal{S}(D_\mathcal{V}))_\mathcal{V}, \vec{m}_T, E = {}^t\vec{m}_T \cdot \vec{Q} + s \cdot Q_0, S = s \cdot Q_0, Y, W = s \cdot Y, \pi_R = \mathcal{S}_R(Q, S, Y, W)\}$ is still perfectly indistinguishable from $\mathcal{D}_2$, and thus from the initial distribution $\mathcal{D}_0$. However, this is clear that no information leaks about individual votes $(\vec{m}_\mathcal{V})_\mathcal{V}$ from $\mathcal{D}_3$, which proves the everlasting privacy. □

We stress that the perfect privacy holds only with respect to the public information in the PBB (even if those values $D_\mathcal{V}$ are associated to the voters $\mathcal{V}$ to provide eligibility verifiability). However, using the commitment randomness $s_\mathcal{V}$, the voter could reveal her vote (from her $D_\mathcal{V}$) and sell it. Hence, to provide receipt-freeness, we will let the server randomize $s_\mathcal{V}$ before publishing the commitment in the PBB. This randomization will not impact the verifiability, as shown below.

Verifiability. In order to be sure that the decryption of $(F_0, \vec{F})$ will lead to a vector $\vec{m}_T$ that is consistent with W, the server must verify the relations between $(C_{\mathcal{V},0}, \vec{C}_\mathcal{V})$ and $D_\mathcal{V}$ with the same $\vec{m}_\mathcal{V}$, and between $D_\mathcal{V}$ and $V_\mathcal{V}$ with the same $s_\mathcal{V}$, from the zero-knowledge proof of consistency $\pi_\mathcal{V}$, for each voter $\mathcal{V}$. But this is just the responsibility of the server, as this proof is not part of the universal verifiability.

For the universal verifiability explained below, we will also need a proof of validity $\mathit{\Pi}_\mathcal{V}$ for $D_\mathcal{V}$, that must be a proof of knowledge of $(\vec{m}_\mathcal{V}, s_\mathcal{V})$ used in $D_\mathcal{V}$. Furthermore, when we target homomorphic tally, we additionally need the guarantee the vote $\vec{m}_\mathcal{V}$ is of the appropriate format (e.g., with 0 or 1, only).

For the former proof $\pi_\mathcal{V}$, we will use basic LH-Sign, without tags. The latter proof of knowledge $\mathit{\Pi}_\mathcal{V}$ will exploit LH-Sign-RTag [HPP20,Poi23], in the case $\vec{m}_\mathcal{V}$ must be proven in a specific set $\mathcal{S}$, as needed for homomorphic tally. Both proofs being perfectly zero-knowledge, they satisfy the requirements of the Theorem 7. For both proofs, knowledge-soundness will rely in the extractable-unforgeability of the signature schemes, which hold under the SXSDL assumption in the Algebraic Group Model, with the above FHS signature.

Theorem 8 (Soundness of the Universal Verifiability). *Given the public information PBB, with the valid proofs of knowledge $(\mathit{\Pi}_\mathcal{V})_\mathcal{V}$, E, $\vec{m}_T$, and S, the proof π_R (with statistical soundness) ensures $\vec{m}_T$ is the result of all the votes committed in the $D_\mathcal{V}$ by each voter, unless one can break the* DL *assumption in the Algebraic Group Model.*

Proof. Thanks to the proofs of knowledge $\mathit{\Pi}_\mathcal{V}$'s, one can extract $(\vec{m}_\mathcal{V}, s_\mathcal{V})$ for each commitment $D_\mathcal{V}$. Then, one can provide an opening $(\vec{m}_T^*, s^*)$ of E: $E = {}^t\vec{m}_T^* \cdot \vec{Q} + s^* \cdot Q_0$. The vector $\vec{m}_T^*$ is the expected result of the election, and we want to show the announced result $\vec{m}_T$ is the same.

Thanks to the statistical soundness of π_R, its validity ensures (Q_0, S, Y, W) is a valid Diffie-Hellman tuple, with overwhelming probability. Then, in the AGM, the algorithm (the voters together with the server) that has generated such a valid $W = y \cdot S$, whereas only $Y = y \cdot Q_0$ is known, is associated to an extractor that outputs s such that $S = s \cdot Q_0$, under the DL assumption: more formally, let us be given a DL instance $(Q_0, Y = y \cdot Q_0)$, an algorithm that is given Y and $S = s \cdot Q_0$, and outputs $W = sy \cdot Q_0$, in the AGM also outputs a linear combination of the inputs: $W = sy \cdot Q_0 = s \cdot Y = a \cdot S + b \cdot Y = as \cdot Q_0 + b \cdot Y$. Which leads to $(s - b) \cdot Y = as \cdot Q_0$. If one would know $s \neq b$, then one could extract $y = as \cdot (s - b)^{-1} \bmod p$. Hence, necessarily, $b = s$ and $a = 0$.

Eventually, $S = s \cdot Q_0 = E - {}^t\vec{m}_T \cdot \vec{Q}$: ${}^t(\vec{m}_T^* - \vec{m}_T) \cdot \vec{Q} = (s^* - s) \cdot Q_0$, for known $s, s^*, \vec{m}_T, \vec{m}_T^*$. If the expected $\vec{m}_T^*$ and the announced $\vec{m}_T$ are different, one can extract the discrete logarithm between some of the components of $\vec{Q}$ and Q_0, that is hard to get under the DL assumption. □

Using our explicit proofs $(\pi, \mathit{\Pi})$ that use LH-Sign and LH-Sign-RTag, with knowledge-soundness under the SXSDL assumption in the AGM as explained below in the section 4.4, and π_R, with statistical soundness in the ROM as shown above in the section 4.2, as the SXSDL assumption implies the DL assumption, we get:

Corollary 9 (Universal Verifiability). *Given the public information* PBB*, with the valid proofs of knowledge* $(\Pi_\mathcal{V})_\mathcal{V}$*,* E*,* $\vec{m}_T$*, and* S*, the proof* π_R *ensures* $\vec{m}_T$ *is the result of all the votes committed in the* $D_\mathcal{V}$ *by each voter, unless one can break the* SXSDL *assumption in the Algebraic Group Model and the Random Oracle Model.*

4.4 Verifiable Ballot under Linear Relations

Let us now explain how one can prove a Verifiable Pedersen commitment/ElGamal ciphertext $(C_0 = r \cdot P, \vec{C} = \vec{m} \cdot P + r \cdot \vec{Z}, D = {}^t\vec{m} \cdot \vec{Q} + s \cdot Q_0, V = s \cdot Y) \in \mathbb{G}^{3+n}$ actually contains an input message $\vec{m} \in \mathcal{S} \subset \mathbb{Z}_p^n$, under the encryption key $\vec{Z} \in \mathbb{G}^n$, and the verification key $Y \in \mathbb{G}$, when the valid set $\mathcal{S}$ can be expressed with K matrices $\mathbf{H}_k \in \mathbb{Z}_p^{\ell_k \times n}$:

$$\vec{m} \in \mathcal{S} \iff \mathbf{H}_k \cdot \vec{m} \cdot P \in \mathcal{S}_k \subseteq \mathbb{G}^{\ell_k}, \text{ for } k = 1, \ldots, K.$$

We of course expect these proofs to be perfectly zero-knowledge, as no information should leak about $\vec{m}$.

Consistency of Pedersen Commitment and ElGamal Ciphertext (π). To get consistency between the ciphertext and the commitment, one wants to check that

$$\begin{aligned}\mathbf{C} &= (C_0 = r \cdot P, \vec{C} = \vec{m} \cdot P + r \cdot \vec{Z}, D = {}^t\vec{m} \cdot \vec{Q} + s \cdot Q_0, V = s \cdot Y) \\ &= r \cdot (P, \vec{Z}, 0, 0) + \sum_i m_i \cdot (0, \vec{e}_{n,i} \cdot P, Q_i, 0) + s \cdot (0, \vec{0}, Q_0, Y)\end{aligned}$$

for some witness $(\vec{m}, r, s)$, where $(\vec{e}_{n,i})_i$ is the canonical basis of $\mathbb{Z}_p^n$. If the authority generates the following $\mathsf{LH\text{-}Sign}$ signatures under a verification key VK for the messages in $\mathbb{G}^{n+3}$:

$$\Sigma_0 = \mathsf{Sign}(\mathsf{SK}, (P, \vec{Z}, 0, 0)), \qquad \Sigma_i = \mathsf{Sign}(\mathsf{SK}, (0, \vec{e}_{n,i} \cdot P, Q_i, 0), \text{ for } i = 1, \ldots, n$$
$$\Sigma_{n+1} = \mathsf{Sign}(\mathsf{SK}, (0, \vec{0}, Q_0, Y)),$$

using the witness $(\vec{m}, r, s)$, the proof can be computed as $\pi = \Sigma = r \cdot \Sigma_0 + \sum_{i=1}^n m_i \cdot \Sigma_i + s \cdot \Sigma_{n+1}$. It consists of a unique group element, and can be checked as $\mathsf{Verif}(\mathsf{VK}, (C_0, \vec{C}, D, V), \pi)$. We stress that we ignore tags Tag and τ, as we consider a unique vector subspace. Under the Extractable Unforgeability of the $\mathsf{LH\text{-}Sign}$, any ciphertext-commitment $\mathbf{C} = (C_0, \vec{C}, D, V)$ that is associated to a valid signature Σ, must be a linear combination (with known coefficients) of the initially signed messages, which provides knowledge-soundness. On the other hand, using the signing key, one can simulate the proof for any tuple $\mathbf{C}$. We can thus claim the following statement, even if a simple zero-knowledge proof of membership would be enough.

Theorem 10 (Zero-Knowledge Proof of Knowledge π). *The above proof* π *is a Zero-Knowledge Proof of Knowledge of* $(\vec{m}, s, r)$ *such that* $\mathbf{C} = (C_0 = r \cdot P, \vec{C} = \vec{m} \cdot P + r \cdot \vec{Z}, D = {}^t\vec{m} \cdot \vec{Q} + s \cdot Q_0, V = s \cdot Y)$*. The knowledge-soundness relies on the extractable unforgeability of the* $\mathsf{LH\text{-}Sign}$*. It is furthermore perfectly zero-knowledge.*

Validity of Ballots (Π). However, the above proof π is for the server only, whereas we also need a proof of knowledge of $(\vec{m}, s)$ in D, for universal verifiability. In case of homomorphic tally, we additionally need to enforce $\vec{m}$ to be a valid vote, in the set $\mathcal{S}$, characterized by the L linear systems $(\mathbf{H}_k \in \mathbb{Z}_p^{\ell_k \times n})_k$, for $k \in \{1, \ldots, K\}$. In the following, we denote $\vec{Q}_{\ell_k}$ a vector of ℓ_k independent generators. As ℓ_k is often smaller than n, it can be the truncation of $\vec{Q}$ to its ℓ_k first components.

Let us illustrate with one system $\mathbf{H}_k \in \mathbb{Z}_p^{\ell_k \times n}$, and thus on the set $\mathcal{S}_k = \{\vec{m}_{k,1}, \ldots, \vec{m}_{k,N_k}\} \subset \mathbb{Z}_p^{\ell_k}$ of acceptable values. One builds a first component P_k as a fixed group element that depends on the relation-set $(\mathbf{H}_k, \mathcal{S}_k)$. It can be set as $P_k = \mathcal{H}(\mathbf{H}_k, \mathcal{S}_k)$ or $P_k = \mathcal{H}(k)$ where the function $\mathcal{H}$ is assumed to be a full-domain hash function that outputs independent group elements for any new query (modelled as a random oracle onto $\mathbb{G}$). The authority generates $\mathsf{LH\text{-}Sign\text{-}RTag}$ signatures under a verification key VK' for messages in $\mathbb{G}^3$, and a tag $\mathsf{Tag}_{k,j}$, for $j = 1, \ldots, N_k$: $\sigma_{k,j,0}$ on $(P_k, 0, -{}^t\vec{m}_{k,j} \cdot \vec{Q}_{\ell_k})$,

$\sigma_{k,j,i}$ on $(0, Q_i, ({}^t\mathbf{H}_k \cdot \vec{Q}_{\ell_k})_i)$, for $i = 1, \ldots, n$, and $\sigma_{k,j,n+1}$ on $(0, Q_0, 0)$. As $\mathbf{H}_k \cdot \vec{m}$ must lie in $\mathcal{S}_k$, this is $\vec{m}_{k,j}$ for some $j \in \{1, \ldots, N_k\}$. From linear properties, we can state the following relation, ${}^t\vec{m}_{k,j} \cdot \vec{Q}_{\ell_k} = {}^t(\mathbf{H}_k \cdot \vec{m}) \cdot \vec{Q}_{\ell_k} = {}^t\vec{m} \cdot ({}^t\mathbf{H}_k \cdot \vec{Q}_{\ell_k})$, then

$$(P_k, 0, -{}^t\vec{m}_{k,j} \cdot \vec{Q}_{\ell_k}) + \sum_i m_i \cdot (0, Q_i, ({}^t\mathbf{H}_k \cdot \vec{Q}_{\ell_k})_i) + s \cdot (0, Q_0, 0)$$
$$= (P_k, {}^t\vec{m} \cdot \vec{Q} + s \cdot Q_0, 0) = (P_k, D, 0)$$

As a consequence, $\tilde{\sigma}_k = \underset{\sim}{\sigma_{k,j,0}} + \sum_{i=1}^n m_i \cdot \sigma_{k,j,i} + s \cdot \underset{\sim}{\sigma_{k,j,n+1}}$ is a valid signature of $(P_k, D, 0) \in \mathbb{G}^3$, under VK' and the tag $\widetilde{\mathsf{Tag}}_k = \mathsf{Tag}_{k,j}$. While the tag $\widetilde{\mathsf{Tag}}_k$ reveals $\vec{m}_{k,j}$, we can exploit the perfect randomizability of $(\widetilde{\mathsf{Tag}}_k, \tilde{\sigma}_k)$ to perfectly hide $\vec{m}_{k,j}$ in both the tag Tag_k and the signature σ_k.

Then, this pair $(\mathsf{Tag}_k, \sigma_k)$ can be defined to be the proof of knowledge of $(\vec{m}, s)$ such that $D = {}^t\vec{m} \cdot \vec{Q} + s \cdot Q_0$, and $\mathbf{H}_k \cdot \vec{m} \cdot P \in \mathcal{S}_k$. The verification checks whether $\mathsf{Verif}(\mathsf{VK}', \mathsf{Tag}_k, (P_k, D, 0), \sigma_k)$ is true or not.

Any such pair that passes the verification contains a valid signature σ_k. Extractable unforgeability provides a linear combination of messages signed with the same tag Tag_{k,j^*}, for some j^*: $(P_k, D, 0) = a \cdot (P_k, 0, -{}^t\vec{m}_{k,j^*} \cdot \vec{Q}_{\ell_k}) + \sum_i m_i \cdot (0, Q_i, ({}^t\mathbf{H}_k \cdot \vec{Q}_{\ell_k})_i) + s \cdot (0, Q_0, 0)$. Necessarily, $a = 0$ and $D = {}^t\vec{m} \cdot \vec{Q} + s \cdot Q_0$. Furthermore, ${}^t\vec{m}_{k,j^*} \cdot \vec{Q}_{\ell_k} = {}^t(\mathbf{H}_k \cdot \vec{m}) \cdot \vec{Q}_{\ell_k}$. If $\mathbf{H}_k \cdot \vec{m} \neq \vec{m}_{k,j^*}$, one gets a non-trivial relation between the components of $\vec{Q}_{\ell_k}$ which can break the DL assumption. This provides knowledge-soundness. On the other hand, using the signing key, one can simulate the proof for any D and any k. Hence, $\Pi = (\mathsf{Tag}_k, \sigma_k)_k$, which consists of $2K$ elements in $\mathbb{G}$ and K elements in $\hat{\mathbb{G}}$, proves the knowledge of $(\vec{m}, s)$ such that $D = {}^t\vec{m} \cdot \vec{Q} + s \cdot Q_0$, and $\vec{m} \in \mathcal{S}$. The smaller K is, the smaller Π will be. We can claim the statement:

Theorem 11 (Zero-Knowledge Proof of Knowledge Π). *The above proof Π is a Zero-Knowledge Proof of Knowledge of $(\vec{m}, s)$ such that $D = {}^t\vec{m} \cdot \vec{Q} + s \cdot Q_0$ and $\vec{m} \in \mathcal{S}$, characterized by the L linear systems $(\mathbf{H}_k \in \mathbb{Z}_p^{\ell_k \times n})_k$, for $k \in \{1, \ldots, K\}$. The knowledge-soundness relies on the extractable unforgeability of the LH-Sign-RTag and the* DL *assumption. It is furthermore perfectly zero-knowledge.*

Secret Ballot-Box and Public Bulleting-Board. The above verifiability requires the server to check both π and Π, for each ballot.

Universal verifiability requires the integrity of the pairs $(\mathcal{V}, D_\mathcal{V})$ and can then check the validity of $(D_\mathcal{V}, \Pi_\mathcal{V} = (\mathsf{Tag}_k, \sigma_k)_k)$ in PBB. The former integrity of the pairs $(\mathcal{V}, D_\mathcal{V})$ is controlled by the individual verifiability of each voter that has kept a fingerprint $\mathcal{H}(D_\mathcal{V})$ of her $D_\mathcal{V}$ to control it has not been altered by the server. Hence, the PBB contains all the tuples $(\mathcal{V}, D_\mathcal{V}, \Pi_\mathcal{V} = (\mathsf{Tag}_k, \sigma_k)_k)$, indexed by $\mathcal{H}(D_\mathcal{V})$ that is the receipt of voter's $\mathcal{V}$, whereas SBB can store on-the-fly aggregation of the ciphertexts $(F_0, \vec{F}, E, W)$, thanks to the guarantee of the consistency between the ciphertexts and the commitments, from the π's, which verifications are the responsibility of the server: if consistency does not hold, the secret y will not help to prove the final result $\vec{m}_T$. Verification will fail.

4.5 Receipt-Freeness and Randomization

Unfortunately, because of the tuples $(\mathcal{V}, D_\mathcal{V}, \Pi_\mathcal{V} = (\mathsf{Tag}_k, \sigma_k)_k)$ in PBB, the knowledge of $s_\mathcal{V}$ can help a voter to prove her vote: $\mathcal{H}(D_\mathcal{V})$ is a receipt to sell her vote. Hence, before storing the ballot in the ballot box, the server must randomize $s_\mathcal{V}$ in $D_\mathcal{V}$: $D'_\mathcal{V} = D_\mathcal{V} + s'_\mathcal{V} \cdot Q_0$, and the server must also adapt the aggregation W with $V'_\mathcal{V} = V_\mathcal{V} + s'_\mathcal{V} \cdot Y$ instead of $V_\mathcal{V}$. But the value $D'_\mathcal{V}$ cannot be checked anymore:

– together with $\Pi_\mathcal{V} = (\mathsf{Tag}_k, \sigma_k)_k$ to verify the format of the vote;
– for the individual verifiability, that ensures the integrity PBB.

The former point is solved with the addition of σ'_k, the signature of $(0, Q_0, 0)$ under Tag_k and VK', so that the server can compute σ''_k, a randomization of $\sigma_k + s'_\mathcal{V} \cdot \sigma'_k$, the signature of $(P_k, D'_\mathcal{V}, 0)$ under Tag'_k (randomized from Tag_k). The proof $\Pi_\mathcal{V}$ is then replaced by $\Pi'_\mathcal{V} = (\mathsf{Tag}'_k, \sigma''_k)_k$, which provides the same guarantees from the universal verifiability point of view.

The latter point is solved in the classical way with a $\mathsf{LH\text{-}Sign}$ verification key $\mathsf{vk}_\mathcal{V} \in \hat{\mathbb{G}}^2$ chosen by the voter $\mathcal{V}$, to sign elements in $\mathbb{G}^2$: $(P, D_\mathcal{V})$ in $\sigma_\mathcal{V}$ and $(0, Q_0)$ in $\sigma'_\mathcal{V}$, so that the server can provide the additional signature $\sigma''_\mathcal{V}$ on $(P, D'_\mathcal{V})$, thanks to the linearity property.

The PBB now contains $(\mathcal{V}, \mathsf{vk}_\mathcal{V}, D'_\mathcal{V}, \Pi'_\mathcal{V} = (\mathsf{Tag}'_k, \sigma''_k)_k, \sigma''_\mathcal{V})$, indexed by the fingerprint $\mathcal{H}(\mathsf{vk}_\mathcal{V})$, the proof of vote kept by the voter, to check $D'_\mathcal{V}$ is an appropriate randomization of $D_\mathcal{V}$, in the individual verifiability, thanks to the extractable unforgeability of the $\mathsf{LH\text{-}Sign}$.

4.6 Global Security

The individual verifiability allows voters to control the integrity of the $D'_\mathcal{V}$'s (no removed bulletins) in the PBB; universal verifiability allows everybody to check the validity of the ballots (well-formed), to compute the aggregation E of all the $D'_\mathcal{V}$, and to eventually check the correctness of the result $\vec{m}_T$ with respect to E. PBB is enough for the overall honest-behavior verification. There is no need to check anything in SBB, except to make sure the result can actually be computed: integrity of SBB will eventually be checked by the validity of the tally w.r.t. E; privacy of the individual ballots will be ensured by the on-the-fly aggregation in $(F_0, \vec{F}, W)$. Once individual ballots have been aggregated and deleted, even a powerful adversary has no way to get any information about the votes.

5 Efficiency

5.1 Complexity and Communications

For an election on n-bit votes, under K conditions defined by the matrices $(\mathbf{H}_k)_k$ of size $\ell_k \leq n$ each, with N_k values in $\mathcal{S}_k$, and for $N = \sum_k N_k$, one first has to publish, as trusted global parameters of the election. Then we detail the cost for the voter.

Global Parameters. The global parameters consists of

- the encryption and verification keys $\vec{Z}$ and Y, where the secret keys $\vec{z}$ and y must be kept secret: $n+1$ elements in $\mathbb{G}$;
- the commitment parameters Q_0 and $\vec{Q}$, that must be generated as independent random generators: $n+1$ elements in $\mathbb{G}$, that can be generated from a random map from $\{0,1\}^*$ to $\mathbb{G}$;
- the verification keys $\mathsf{VK} \in \hat{\mathbb{G}}^{n+3}$ and $\mathsf{VK}' \in \hat{\mathbb{G}}^3$, where the secret signing keys are just kept during the initialization phase to generate the signatures below: $n+6$ elements in $\hat{\mathbb{G}}$;
- the signatures Σ_i, for $i = 0, \ldots, n+1$: $n+2$ elements in $\mathbb{G}$;
- the tags $\mathsf{Tag}_{k,j}$ and the signatures $\sigma_{k,j,i}$, for $i = 0, \ldots, n+1$: $n+3$ elements in $\mathbb{G}$ and 1 element in $\hat{\mathbb{G}}$, for $k = 1, \ldots, K$, and $j = 1, \ldots, N_k$.

The global parameters contain $(N+3)(n+3) - 5$ elements in $\mathbb{G}$, and $N + n + 6$ elements in $\hat{\mathbb{G}}$.

Distributed Generation. We stress they can all be generated in a distributed way to avoid relying on a single truster party. The full generation of the global parameters is described and formally proven in the Appendix B, but we present here the most complex one, which is the generation of the the linearly-homomorphic signatures with randomizable tags. We consider the 2-party case, with $\mathcal{P}_1$ and $\mathcal{P}_2$ that do not collude. All the tags, indexed by t, and all the messages $\vec{M}_{t,i}$, are dealt in parallel, with $R_{t,i} = \mathcal{H}(\mathsf{Election}, \text{“Signatures”}, t, i) \in \mathbb{G}$:

- Client $\mathcal{P}_1$
  - chooses $\mathsf{SK}'_1 \xleftarrow{\$} \mathbb{Z}_p^3$ and computes $\mathsf{VK}'_1 = \mathsf{SK}'_1 \cdot \hat{P}$;
  - chooses $(\alpha_{t,1} \xleftarrow{\$} \mathbb{Z}_p)_t$ and computes $(\sigma_{t,i,1} = \alpha_{t,1} \cdot ({}^t\mathsf{SK}'_1 \cdot \vec{M}_{t,i} + R_{t,i}))_{t,i}$;
  - sends VK'_1 and $(\sigma_{t,i,1})_{t,i}$ to $\mathcal{P}_2$;
- Client $\mathcal{P}_2$
  - chooses $\mathsf{SK}'_2 \xleftarrow{\$} \mathbb{Z}_p^3$ and computes $\mathsf{VK}'_2 = \mathsf{SK}'_2 \cdot \hat{P}$;
  - chooses $(\alpha_{t,2} \xleftarrow{\$} \mathbb{Z}_p)_t$ and computes $(\sigma_{t,i,2} = \alpha_{t,2} \cdot ({}^t\mathsf{SK}'_2 \cdot \vec{M}_{t,i} - R_{t,i}))_{t,i}$ and $(\sigma'_{t,i,1} = \alpha_{t,2} \cdot \sigma_{t,i,1})_{t,i}$;
  - sends VK'_2 and $(\sigma_{t,i,2}, \sigma'_{t,i,1})_{t,i}$ to $\mathcal{P}_1$;
- Client $\mathcal{P}_1$
  - chooses $(\tau_{t,1} \xleftarrow{\$} \mathbb{Z}_p)_t$ and computes $(Q_{t,1} = 1/\tau_{t,1} \cdot P, \hat{Q}_{t,1} = 1/\tau_{t,1} \cdot \hat{P})_t$ and $(\sigma'_{t,i} = \tau_{t,1} \cdot (\sigma_{t,i,2} + 1/\alpha_{t,1} \cdot \sigma'_{t,i,1}))_{t,i}$;
  - sends $(\sigma'_{t,i})_{t,i}$ and $(Q_{t,1}, \hat{Q}_{t,1})_t$ to $\mathcal{P}_2$;

Public Parameters # elements in		Voter Computation # scalar multiplications in		Ballot Size # elements in	
$\mathbb{G}$	$\hat{\mathbb{G}}$	$\mathbb{G}$	$\hat{\mathbb{G}}$	$\mathbb{G}$	$\hat{\mathbb{G}}$
$(N+3)(n+3)-5$	$N+n+6$	$4K+n+8$	$K+2$	$3K+n+7$	$K+2$

Fig. 1. Complexity of the Global Process, with Public Parameters including $(\vec{Z}, Y, Q, \vec{Q}, \mathsf{VK}, \mathsf{VK}')$ and the signatures with tags $(\Sigma_i)_i$, $(\mathsf{Tag}_{k,j}, (\sigma_{k,j,i})_i)_{k,j}$, and the ballots contain the randomization signatures for receipt-freeness.

- Client $\mathcal{P}_2$
  - chooses $(\tau_{t,2} \xleftarrow{\$} \mathbb{Z}_p)_t$ and computes $(Q_t = 1/\tau_{t,2} \cdot Q_{t,1}, \hat{Q}_t = 1/\tau_{t,2} \cdot \hat{Q}_{t,1})_t$ and $(\sigma_{t,i} = \tau_{t,2}/\alpha_{t,2} \cdot \sigma'_{t,i})_{t,i}$;
  - sends $(\sigma_{t,i})_{t,i}$ and $(Q_t, \hat{Q}_t)_t$ to $\mathcal{P}_1$;
- Everybody
  - computes $\mathsf{VK}' = \mathsf{VK}'_1 + \mathsf{VK}'_2$;
  - stores $(Q_t, \hat{Q}_t)_t$ and $(\sigma_{t,i})_{t,i}$;
  - checks $\mathsf{LH\text{-}Sign\text{-}RTag.Verif}(\mathsf{VK}', (Q_t, \hat{Q}_t), \vec{M}_{t,i}, \sigma_{t,i})$, for all t and i.

We can prove (see the Appendix B) that no additional information leaks beyond the signatures on the fixed messages with random tags under a random verification key VK', to a semi-honest adversary (honest-but-curious), unless one can break the DDH assumption in $\mathbb{G}$. We stress this is important for the soundness of the validity proofs, but it does not impact the privacy property, that is still unconditional.

Ballot Generation. To produce her vote, the voter has to compute and publish:
- the tuple $\mathbf{C} = (C_0 = r \cdot P, \vec{C} = \vec{m} \cdot P + r \cdot \vec{Z}, D = {}^t\vec{m} \cdot \vec{Q} + s \cdot Q_0, V = s \cdot Y)$: $n+3$ scalar multiplications in $\mathbb{G}$ to send $n+3$ elements in $\mathbb{G}$, as $\vec{m} \in \{0,1\}^n$;
- the signature Σ on $\mathbf{C}$: 2 scalar multiplications in $\mathbb{G}$ to send 1 element in $\mathbb{G}$;
- the signatures σ_k on $(P_k, D, 0)$ with the randomized tags Tag_k, for $k = 1, \dots, K$: $3K$ scalar multiplications in $\mathbb{G}$ and K scalar multiplications in $\hat{\mathbb{G}}$ to send $2K$ element in $\mathbb{G}$ and K element in $\hat{\mathbb{G}}$.
- for additional receipt-freeness, the signature σ'_k of $(0, Q_0, 0)$ under the randomized tag Tag_k, and a verification key $\mathsf{vk}_{\mathcal{V}} \in \hat{\mathbb{G}}^2$, with the signatures $\sigma_{\mathcal{V}}$ and $\sigma'_{\mathcal{V}}$ of (P, D) and $(0, Q_0)$ respectively.

The voter performs $4K+n+8$ scalar multiplications in $\mathbb{G}$ and $K+2$ scalar multiplications in $\hat{\mathbb{G}}$; and sends $3K+n+7$ elements in $\mathbb{G}$ and $K+2$ element in $\hat{\mathbb{G}}$. The global numbers are recalled in Figure 1.

5.2 Examples of Elections

The 1-out-of-n Votes is very classical, for votes $\vec{m} \in \{0,1\}^n$ with the additional condition $\sum m_i \in \{0,1\}$: $\mathbf{H}_k = \left[\vec{e}_{n,k}\right]$, for $\ell_k = 1$, with $\mathcal{S}_k = \{0,1\}$, for $k = 1, \dots, n$ and $\mathbf{H}_{n+1} = \left[1\ 1 \dots 1\right]$, for $\ell_{n+1} = 1$ with $\mathcal{S}_{n+1} = \{0,1\}$: $K = n+1$ and $N = 2n+2$.

This is the classical approach for such a vote: one proves each box to be 0 or 1, and then the sum is also 0 or 1, as one does with Schnorr-like proofs. But then $n+1$ proofs are needed: this leads to a costly and large ballot. With our particular proof technique, the generation complexity and the final size of the proof can be much smaller, using $\mathbf{H} = \mathbf{Id}_n$ and $\mathcal{S} = \{\vec{0}_n, \vec{e}_{n,1}, \dots, \vec{e}_{n,n}\}$, for $\ell = n$, as $N = n+1$ and $K = 1$, see Figure 2 with the former basic (B) approach and the latter optimized (O) approach.

The t-out-of-n Votes can be for votes $\vec{m} \in \{0,1\}^n$ with the additional condition $\sum m_i \in \{0, \dots, t\}$: $\mathbf{H}_k = \left[\vec{e}_{n,k}\right]$, for $\ell_k = 1$, with $\mathcal{S}_k = \{0, P\}$, for $k = 1, \dots, n$ and $\mathbf{H}_{n+1} = \left[1\ 1 \dots 1\right]$, for $\ell_{n+1} = 1$ with $\mathcal{S}_{n+1} = \{0, 1, \dots, t\}$: $K = n+1$ and $N = 2n+t+1$.

This is again the classical approach: one proves each box to be 0 or 1, and then the sum is at most t, with $n+1$ proofs. We can reduce the cost, with trade-offs, using $\mathbf{H} = \mathbf{Id}_n$ and $\mathcal{S}$ is the set of all the possible votes, with consists of $\binom{n}{t} \leq n^t$ values: $K = 1$ and $N \leq n^t$. A large N only impacts the size of the global parameters, but does not impact the generation of the ballots, as there is essentially the encryption of $\vec{m}$ (which is definitely required) and very few additional multiplications and group elements. This is optimal from the voter's point of view!

If n^t becomes too large for a reasonable size of the global parameters, many trade-offs are possible: with $n = a \times b$, one can set $\mathbf{H}_k = \mathbf{0}_{b\times(k-1)b}||\mathbf{Id}_b||\mathbf{0}_{b\times(a-k)b}$ and $\mathcal{S}_k$ is all the 2^b possible vectors in $\{0,1\}^b$, with $\ell_k = b$, for $k = 1,\ldots,a$, and $\mathbf{H}_{a+1} = \begin{bmatrix}1\ 1\ldots 1\end{bmatrix}$, for $\ell_{a+1} = 1$, with $\mathcal{S}_{a+1} = \{0,1,\ldots,t\}$: $K = a+1$ and $N = a \times 2^b + t + 1$. Again, the smaller a is, the more efficient it is for the voter.

List Voting with Deletion is wildly used in France, where one can choose at most one list of candidates, and delete some of the candidates on the chosen list. This leads to very complex constraints, that are hard to be verified with classical approaches.

Let us consider the case with T lists of R_t candidates each. The vote can be expressed as a vector $\vec{m} = (u_1, v_{1,1}, \ldots, v_{i,R_i}, \ldots, u_T, v_{T,1}, \ldots, v_{T,R_T}) \in \{0,1\}^{T+R}$, where $R = \sum_t R_t$: $(u_t)_t$ declares which is the chosen list, so $(u_t)_t$ contains at most one component to one, and $(v_{t,j})_{t,j}$ are the chosen candidates, where $(v_{t,j})_j$ is not all zero if $u_t = 1$, for any $t \in \{1,\ldots,T\}$. The constraints are indeed

- each box is selected or not: for any $t \in \{1,\ldots,T\}$, $u_t \in \{0,1\}$ and $v_{t,j} \in \{0,1\}$, for $j = 1,\ldots,R_t$;
- at most one list is selected: $(u_t)_t \in \{\vec{0}, \vec{e}_{T,1}, \ldots, \vec{e}_{T,T}\}$;
- as soon as at least one of the candidates in a list is selected, this list is selected too: for any $t \in \{1,\ldots,T\}$, $(u_t, \sum_j v_{t,j}) \in \{(0,0),(1,1),\ldots,(1,R_t)\}$.

All these relations can be compressed into $K = R + T + 1$ linear constraints:

$$v_{t,j} \in \{0,1\}, \text{for } t = 1,\ldots,T \text{ and } j = 1,\ldots,R_t \qquad \sum_t u_t \in \{0,1\}$$

$$(u_t, \sum_j v_{t,j}) \in \{(0,0),(1,1),\ldots,(1,R_t)\}, \text{for } t = 1,\ldots,T$$

and they can be verified with

$$\mathbf{H}_{t,j} = \begin{pmatrix} 0 & \vec{0}_{R_1} & \cdots & 0 & \vec{0}_{R_{t-1}} & 0 & \vec{e}_{R_t,j} & 0 & \vec{0}_{R_{t+1}} & \cdots & 0 & \vec{0}_{R_T} \end{pmatrix} \qquad \mathbf{H}_{t,j}\cdot\vec{m} \in \mathcal{S}_0,$$
$$\text{for } t = 1,\ldots,T \text{ and } j = 1,\ldots,R_t$$
$$\mathbf{H}_0 = \begin{pmatrix} 1 & \vec{0}_{R_1} & \cdots & 1 & \vec{0}_{R_t} & \cdots & 1 & \vec{0}_{R_T} \end{pmatrix} \qquad \mathbf{H}_0\cdot\vec{m} \in \mathcal{S}_0$$
$$\mathbf{H}_t = \begin{pmatrix} 0 & \vec{0}_{R_1} & \cdots & 0 & \vec{0}_{R_{t-1}} & 1 & \vec{0}_{R_t} & 0 & \vec{0}_{R_{t+1}} & \cdots & 0 & \vec{0}_{R_T} \\ 0 & \vec{0}_{R_1} & \cdots & 0 & \vec{0}_{R_{t-1}} & 0 & \vec{1}_{R_t} & 0 & \vec{0}_{R_{t+1}} & \cdots & 0 & \vec{0}_{R_T} \end{pmatrix} \qquad \mathbf{H}_t\cdot\vec{m} \in \mathcal{S}_t$$
$$\text{for } t = 1,\ldots,T$$

where $\mathcal{S}_0 = \{0,1\}$ and $\mathcal{S}_t = \left\{\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}1\\1\end{pmatrix}, \ldots, \begin{pmatrix}1\\R_t\end{pmatrix}\right\}$, so $N = 3R + T + 2$.

The first relations with matrices $\mathbf{H}_{t,j}$ are just to ensure 0 or 1 choices for $v_{t,j}$. The same trade-offs as above can be exploited to reduce both the voter's computational cost and the ballot size, as illustrated below in Figure 3, where P is the size of the packets (subvectors that are proven in $\{0,1\}^P$).

5.3 Benchmarks

We have implemented a basic proof of concept of the full protocol, with the type III pairing-friendly curve BLS12-381 [BLS03] ($\mathbb{G}$ group elements are encoded on 48 bytes, while $\hat{\mathbb{G}}$ group elements are encoded on 96 bytes) and the Rust library (https://github.com/zkcrypto/bls12_381).

Figure 2 presents the classical single-member constituency, where one votes for at most one candidate (the 1-out-of-n choice), with the 2 types of approaches (the basic (B) approach and the optimized (O) approach). Figure 3 illustrates timings and sizes for list voting with deletion. We also apply ballot randomization before extraction to achieve receipt-freeness and show how trade-offs can be found with our proof technique, and various packet sizes.

In the latter case, with most extreme parameters, for 10 lists of 20 candidates each, while the public parameters are large, the ballot is still quite reasonable, in size and for generation time: less than 1 second for a 25kB ballot. This only depends on the number of relations to prove.

Size		Public	Ballot			PBB	Tally		
n	T	Size	Size	Generate	Extract	Verify	Decrypt	Proof	Verify
25	B	96kB	8.83kB	93ms	209ms	391ms	4ms	13ms	407ms
	O	53kB	2.19kB	39ms	102ms	24ms			38ms
50	B	327kB	16.84kB	294ms	369ms	944ms	5ms	25ms	786ms
	O	178kB	3.55kB	67ms	149ms	25ms			50ms

Fig. 2. Benchmarks for a 1-out-of-n choice (B, is for the basic approach, and O for the optimized approach), with 5 bulletins in the tally phase, on a Macbook Pro M1 14in.

Size			Public	Ballot			PBB	Tally		
T	C	P	Size	Size	Generate	Extract	Verify	Decrypt	Proof	Verify
5	10	1	533kB	18.2kB	309ms	390ms	821ms	6ms	28ms	878ms
5	10	5	1.23MB	7.5kB	119ms	206ms	231ms			268ms
5	10	10	16.82MB	6.2kB	99ms	183ms	158ms			186ms
10	10	1	1.96MB	35.8kB	1.07s	744ms	1.63s	12ms	55ms	1.72s
10	10	5	4.67MB	14.5kB	357ms	384ms	463ms			515ms
10	20	1	7.12MB	68kB	3.72s	1.45s	3.16s	18ms	102ms	3.19s
10	20	4	11.72MB	28kB	1.16s	710ms	948ms			1.05s
10	20	5	17.25MB	25kB	975ms	651ms	773ms			917ms

Fig. 3. Benchmarks for a List Voting with Deletion (T lists with C candidates each, and thus $R = TC$, using packets of size P), with 5 bulletins in the tally phase, on a Macbook Pro M1 14in.

We stress that these trade-offs are impossible with other classical approaches: the Schnorr-like proofs have both generation time and size at least linear in N and SNARKs have generation time at least linear in N. With the above approach, generation time and size are completely independent of N, which make them quite appropriate in electronic voting, where the client is the most limited party.

Acknowledgments

This work was supported in part by the France 2030 ANR Project ANR-22-PECY-003 SecureCompute.

References

ABC$^+$12. Jae Hyun Ahn, Dan Boneh, Jan Camenisch, Susan Hohenberger, abhi shelat, and Brent Waters. Computing on authenticated data. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 1–20. Springer, Heidelberg, March 2012.

AFG$^+$10. Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 209–236. Springer, Heidelberg, August 2010.

AHO10. Masayuki Abe, Kristiyan Haralambiev, and Miyako Ohkubo. Signing on elements in bilinear groups for modular protocol design. Cryptology ePrint Archive, Report 2010/133, 2010. https://eprint.iacr.org/2010/133.

BCCT13. Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for SNARKS and proof-carrying data. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 111–120. ACM Press, June 2013.

BF11a. Dan Boneh and David Mandell Freeman. Homomorphic signatures for polynomial functions. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 149–168. Springer, Heidelberg, May 2011.

BF11b. Dan Boneh and David Mandell Freeman. Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 1–16. Springer, Heidelberg, March 2011.

BF20. Balthazar Bauer and Georg Fuchsbauer. Efficient signatures on randomizable ciphertexts. In Clemente Galdi and Vladimir Kolesnikov, editors, *SCN 20*, volume 12238 of *LNCS*, pages 359–381. Springer, Heidelberg, September 2020.

BF24. Balthazar Bauer and Georg Fuchsbauer. On security proofs of existing equivalence class signature schemes. Cryptology ePrint Archive, Paper 2024/183, 2024.

BFKW09. Dan Boneh, David Freeman, Jonathan Katz, and Brent Waters. Signing a linear subspace: Signature schemes for network coding. In Stanislaw Jarecki and Gene Tsudik, editors, *PKC 2009*, volume 5443 of *LNCS*, pages 68–87. Springer, Heidelberg, March 2009.

BLS03. Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. Constructing elliptic curves with prescribed embedding degrees. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02*, volume 2576 of *LNCS*, pages 257–267. Springer, Heidelberg, September 2003.

BR93. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993.

CCFG16. Pyrros Chaidos, Véronique Cortier, Georg Fuchsbauer, and David Galindo. BeleniosRF: A non-interactive receipt-free electronic voting scheme. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 1614–1625. ACM Press, October 2016.

CFL19. Véronique Cortier, Alicia Filipiak, and Joseph Lallemand. BeleniosVS: Secrecy and verifiability against a corrupted voting device. In Stephanie Delaune and Limin Jia, editors, *CSF 2019 Computer Security Foundations Symposium*, pages 367–381. IEEE Computer Society Press, 2019.

CPP13. Edouard Cuvelier, Olivier Pereira, and Thomas Peters. Election verifiability or ballot privacy: Do we need to choose? In Jason Crampton, Sushil Jajodia, and Keith Mayes, editors, *ESORICS 2013*, volume 8134 of *LNCS*, pages 481–498. Springer, Heidelberg, September 2013.

CS11. Véronique Cortier and Ben Smyth. Attacking and fixing helios: An analysis of ballot secrecy. In Michael Backes and Steve Zdancewic, editors, *CSF 2011 Computer Security Foundations Symposium*, pages 297–311. IEEE Computer Society Press, 2011.

ElG85. Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

FHS19. Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. *Journal of Cryptology*, 32(2):498–546, April 2019.

FKL18. Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Heidelberg, August 2018.

Fre12. David Mandell Freeman. Improved security for linearly homomorphic signatures: A generic framework. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 697–714. Springer, Heidelberg, May 2012.

GGPR13. Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, Heidelberg, May 2013.

Gro16. Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016.

HMMP23. Thomas Haines, Rafieh Mosaheb, Johannes Müller, and Ivan Pryvalov. SoK: Secure E-voting with everlasting privacy. *PoPETs*, 2023(1):279–293, January 2023.

HP22. Chloé Hébant and David Pointcheval. Traceable Constant-Size Multi-authority Credentials. In Clemente Galdi and Stanislaw Jarecki, editors, *SCN 2022 Security and Cryptography for Networks*, volume 13409 of *LNCS*, pages 411–434. Springer, 2022.

HPP20. Chloé Hébant, Duong Hieu Phan, and David Pointcheval. Linearly-homomorphic signatures and scalable mix-nets. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 597–627. Springer, Heidelberg, May 2020.

JMSW02. Robert Johnson, David Molnar, Dawn Xiaodong Song, and David Wagner. Homomorphic signature schemes. In Bart Preneel, editor, *CT-RSA 2002*, volume 2271 of *LNCS*, pages 244–262. Springer, Heidelberg, February 2002.

LPJY13. Benoît Libert, Thomas Peters, Marc Joye, and Moti Yung. Linearly homomorphic structure-preserving signatures and their applications. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 289–307. Springer, Heidelberg, August 2013.

Pai99. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238. Springer, Heidelberg, May 1999.

Ped92. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, Heidelberg, August 1992.

Poi23. David Pointcheval. Linearly-Homomorphic Signatures for Short Randomizable Proofs of Subset Membership. Cryptology ePrint Archive, Paper 2023/1499, 2023.

A Proof of the Extractable Unforgeability

In this section, we detail the proof of the Theorem 5, on Extractable Unforgeability of the FHS signature scheme as an $\mathsf{LH\text{-}Sign\text{-}RTag}$. In the selective-message scenario, the adversary first outputs K lists of messages $(\vec{M}_{k,j})_{k,j}$, for $k = 1, \dots, K$, and $j = 1, \dots, J_k$: in the AGM, they all come with the scalars $\vec{m}_{k,j}$ such that $\vec{M}_{k,j} = \vec{m}_{k,j} \cdot P$; the challenger generates a random pair of keys $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Keygen}(\mathsf{param}, n)$, as well as K random verifiable tags $(\mathsf{Tag}_k = (Q_k, \hat{Q}_k), \tau_k) \leftarrow \mathsf{NewTag}(\mathsf{param})$, for $k = 1, \dots, K$, and the signatures $\sigma_{k,j} \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{Tag}_k, \tau_k, \vec{M}_{k,j})$, for $k = 1, \dots, K$, and $j = 1, \dots, J_k$;

from $(P, \hat{P})$, $\mathsf{vk} = (\hat{P}_i)_i$, $(\mathsf{Tag}_k = (Q_k, \hat{Q}_k))_k$, and $(\sigma_{k,j})_{k,j})$, the adversary eventually outputs a new tuple $(\mathsf{Tag}^* = (Q^*, \hat{Q}^*), \vec{M}^*, \sigma^*)$.

We will use index $i = 1, \ldots, n$, for enumerating on the components of the vectors; index $k = 1, \ldots, K$, for enumerating on the lists of messages, and index $j = 1, \ldots, J_k$, for enumerating messages $(\vec{M}_{k,j})_j$ into each list. The algebraic adversary also outputs linear combinations, in the corresponding groups $\mathbb{G}$ or $\hat{\mathbb{G}}$:

$$\hat{Q}^* = \alpha \cdot \hat{P} + \sum_i \beta_i \cdot \hat{P}_i + \sum_k \gamma_k \cdot \hat{Q}_k \qquad \begin{aligned} Q^* &= \delta \cdot P + \sum_k \epsilon_k \cdot Q_k + \sum_{k,j} \phi_{k,j} \cdot \sigma_{k,j} \\ M_i^* &= \rho_i \cdot P + \sum_k \zeta_{i,k} \cdot Q_k + \sum_{k,j} \kappa_{i,k,j} \cdot \sigma_{k,j} \\ \sigma^* &= \psi \cdot P + \sum_k \mu_k \cdot Q_k + \sum_{k,j} \nu_{k,j} \cdot \sigma_{k,j} \end{aligned}$$

They must satisfy both $e(P, \hat{Q}^*) = e(Q^*, \hat{P})$ and $e(\sigma^*, \hat{Q}^*) = \prod_i e(M_i^*, \hat{P}_i)$.

Simplifications of the Formula. In a first step, we specify the notations. To this aim, we assume to be given $(G, x \cdot G, y \cdot G, U = xy \cdot G, x \cdot U, y \cdot U, V = xy \cdot U = x^2y^2 \cdot G, \hat{G}, x \cdot \hat{G}, y \cdot \hat{G}, \hat{U} = xy \cdot \hat{G}, x \cdot \hat{U}, y \cdot \hat{U})$, as well as $s_i \cdot U$, $xs_i \cdot U$, $ys_i \cdot U$, and $\hat{P}_i = s_i \cdot \hat{U}$, for all i, for some scalars x, y, and $\vec{s} = (s_i)_i$. We set the generators $P := U$, $\hat{P} := \hat{U}$, and the first tag $(Q_1 := y \cdot G, \hat{Q}_1 := y \cdot \hat{G})$, which means that $\tau_1 = x$, while the other tags are $(Q_k := x/\tau'_k \cdot G, \hat{Q}_1 := x/\tau'_k \cdot \hat{G})$, which means that $\tau_k = y\tau'_k$, for random τ'_k. When only x is unknown, with an SXSDL instance $(G, x \cdot G, x^2 \cdot G, \hat{G}, x \cdot \hat{G})$, one can generate all the elements with known y and $\vec{s}$. It is then hard to extract x. Similarly, for only y unknown, with an SXSDL instance, this is hard to recover y. On the other hand, when only one s_i is unknown, from an SXDL instance $(s_i \cdot P, s_i \cdot \hat{P})$, this is hard to recover s_i:

$$\begin{aligned} \hat{Q}_1 &= 1/x \cdot \hat{P} = y \cdot \hat{G} \\ \\ \hat{Q}_k &= 1/y\tau'_k \cdot \hat{P} = x/\tau'_k \cdot \hat{G} \end{aligned} \qquad \begin{aligned} Q_1 &= 1/x \cdot P = y \cdot G \\ \sigma_{1,j} &= x \cdot ({}^t\vec{m}_{1,j} \cdot \vec{s}) \cdot P = ({}^t\vec{m}_{1,j} \cdot \vec{s}) \cdot x^2y \cdot G \\ Q_k &= 1/y\tau'_k \cdot P = x/\tau'_k \cdot G \\ \sigma_{k,j} &= y\tau'_k \cdot ({}^t\vec{m}_{k,j} \cdot \vec{s}) \cdot P = \tau'_k \cdot ({}^t\vec{m}_{k,j} \cdot \vec{s}) \cdot xy^2 \cdot G \end{aligned}$$

which leads to

$$\begin{aligned} \hat{Q}^* &= \alpha \cdot \hat{U} + \sum_i \beta_i \cdot \hat{P}_i + \gamma_1 \cdot y \cdot \hat{G} + \sum_{k>1} x\gamma_k/\tau'_k \cdot \hat{G} \\ &= \left(\alpha \cdot xy + \sum_i \beta_i \cdot s_i \cdot xy + \gamma_1 \cdot y + \sum_{k>1} x\gamma_k/\tau'_k \right) \cdot \hat{G} \\ Q^* &= \delta \cdot U + \epsilon_1 \cdot y \cdot G + \sum_{k>1} \epsilon_k/\tau'_k \cdot x \cdot G + \sum_j \phi_{1,j} \cdot ({}^t\vec{m}_{1,j} \cdot \vec{s}) \cdot x^2y \cdot G + \sum_{k>1,j} \phi_{k,j} \cdot \tau'_k \cdot ({}^t\vec{m}_{k,j} \cdot \vec{s}) \cdot xy^2 \cdot G \\ &= \Big(\delta \cdot xy + \epsilon_1 \cdot y + \sum_{k>1} \epsilon_k/\tau'_k \cdot x + \sum_j \phi_{1,j} \cdot ({}^t\vec{m}_{1,j} \cdot \vec{s}) \cdot x^2y + \sum_{k>1,j} \phi_{k,j} \cdot \tau'_k \cdot ({}^t\vec{m}_{k,j} \cdot \vec{s}) \cdot xy^2 \Big) \cdot G \end{aligned}$$

and

$$\begin{aligned}
M_i^* &= \rho_i \cdot U + \zeta_{1,i} \cdot y \cdot G + \sum_{k>1} \zeta_{k,i}/\tau'_k \cdot x \cdot G + \sum_j \kappa_{1,j,i} \cdot ({}^t\vec{m}_{1,j} \cdot \vec{s}) \cdot x^2 y \cdot G \\
&\quad + \sum_{k>1,j} \kappa_{k,j,i} \cdot \tau'_k \cdot ({}^t\vec{m}_{k,j} \cdot \vec{s}) \cdot xy^2 \cdot G \\
&= \Big(\rho_i \cdot xy + \zeta_{1,i} \cdot y + \sum_{k>1} \zeta_{k,i}/\tau'_k \cdot x + \sum_j \kappa_{1,j,i} \cdot ({}^t\vec{m}_{1,j} \cdot \vec{s}) \cdot x^2 y \\
&\quad + \sum_{k>1,j} \kappa_{k,j,i} \cdot \tau'_k \cdot ({}^t\vec{m}_{k,j} \cdot \vec{s}) \cdot xy^2\Big) \cdot G \\
\sigma^* &= \psi \cdot U + \mu_1 \cdot y \cdot G + \sum_{k>1} \mu_k/\tau'_k \cdot x \cdot G + \sum_j \nu_{1,j} \cdot ({}^t\vec{m}_{1,j} \cdot \vec{s}) \cdot x^2 y \cdot G \\
&\quad + \sum_{k>1,j} \nu_{k,j} \cdot \tau'_k \cdot ({}^t\vec{m}_{k,j} \cdot \vec{s}) \cdot xy^2 \cdot G \\
&= \Big(\psi \cdot xy + \mu_1 \cdot y + \sum_{k>1} \mu_k/\tau'_k \cdot x + \sum_j \nu_{1,j} \cdot ({}^t\vec{m}_{1,j} \cdot \vec{s}) \cdot x^2 y \\
&\quad + \sum_{k>1,j} \nu_{k,j} \cdot \tau'_k \cdot ({}^t\vec{m}_{k,j} \cdot \vec{s}) \cdot xy^2\Big) \cdot G
\end{aligned}$$

which can be simplified, with known scalars, from the extractor:

$$a = \sum_{k>1} \gamma_k/\tau'_k \qquad \vec{a} = (\beta_i)_i \qquad c = \sum_{k>1} \epsilon_k/\tau'_k$$

$$\vec{c} = \sum_{k>1,j} \phi_{k,j} \cdot \tau'_k \cdot \vec{m}_{k,j} \qquad \vec{d} = \sum_j \phi_{1,j} \cdot \vec{m}_{1,j}$$

into

$$\begin{aligned}
\hat{Q}^* &= \big(\alpha \cdot xy + {}^t\vec{a} \cdot \vec{s} \cdot xy + \gamma_1 \cdot y + a \cdot x\big) \cdot \hat{G} \\
Q^* &= \Big(\delta \cdot xy + \epsilon_1 \cdot y + c \cdot x + {}^t\vec{d} \cdot \vec{s} \cdot x^2 y + {}^t\vec{c} \cdot \vec{s} \cdot xy^2\Big) \cdot G
\end{aligned}$$

and

$$u'_i = \sum_{k>1} \zeta_{k,i}/\tau'_k \qquad \vec{u}_i = \sum_{k>1,j} \kappa_{k,j,i} \cdot \tau'_k \cdot \vec{m}_{k,j} \qquad \vec{v}_i = \sum_j \kappa_{1,j,i} \cdot \vec{m}_{1,j}$$

$$u = \sum_{k>1} \mu_k/\tau'_k \qquad \vec{u} = \sum_{k>1,j} \nu_{k,j} \cdot \tau'_k \cdot \vec{m}_{k,j} \qquad \vec{v} = \sum_j \nu_{1,j} \cdot \vec{m}_{1,j}$$

into

$$\begin{aligned}
M_i^* &= \big(\rho_i \cdot xy + \zeta_{1,i} \cdot y + u'_i \cdot x + {}^t\vec{u}_i \cdot \vec{s} \cdot xy^2 + {}^t\vec{v}_i \cdot \vec{s} \cdot x^2 y\big) \cdot G \\
\sigma^* &= \big(\psi \cdot xy + \mu_1 \cdot y + u \cdot x + {}^t\vec{u} \cdot \vec{s} \cdot xy^2 + {}^t\vec{v} \cdot \vec{s} \cdot x^2 y\big) \cdot G
\end{aligned}$$

Analysis of the New Tag. We now target the new tag involved in the forgery, which satisfies the relation

$$\begin{aligned}
e(P, \hat{Q}^*) &= e(xy \cdot G, \big(\alpha \cdot xy + {}^t\vec{a} \cdot \vec{s} \cdot xy + \gamma_1 \cdot y + a \cdot x\big) \cdot \hat{G}) \\
&= \big(\alpha \cdot x^2y^2 + {}^t\vec{a} \cdot \vec{s} \cdot x^2y^2 + \gamma_1 \cdot xy^2 + a \cdot x^2 y\big) \cdot e(G, \hat{G}) \\
= e(Q^*, \hat{P}) &= e(\Big(\delta \cdot xy + \epsilon_1 \cdot y + c \cdot x + {}^t\vec{d} \cdot \vec{s} \cdot x^2 y + {}^t\vec{c} \cdot \vec{s} \cdot xy^2\Big) \cdot G, xy \cdot \hat{G}) \\
&= \Big(\delta \cdot x^2y^2 + \epsilon_1 \cdot xy^2 + c \cdot x^2 y + {}^t\vec{d} \cdot \vec{s} \cdot x^3y^2 + {}^t\vec{c} \cdot \vec{s} \cdot x^2y^3\Big) \cdot e(G, \hat{G})
\end{aligned}$$

which means, in basis $e(G,\hat{G})$, using the scalars x, y and $\vec{s}$, in $\mathbb{Z}_p$:

$$\begin{aligned}\alpha\cdot x^2y^2 + {}^t\vec{a}\cdot\vec{s}\cdot x^2y^2 + \gamma_1\cdot xy^2 + a\cdot x^2y\\ = \delta\cdot x^2y^2 + \epsilon_1\cdot xy^2 + c\cdot x^2y + {}^t\vec{d}\cdot\vec{s}\cdot x^3y^2 + {}^t\vec{c}\cdot\vec{s}\cdot x^2y^3\end{aligned}$$

Knowing all the s_i's, and y, with only the SXSDL challenge $(G, x\cdot G, x^2\cdot G, \hat{G}, x\cdot\hat{G})$ for $x\neq 0$, if the relation is not trivial, with non-negligible probability, one can solve the quadratic equation and find x. Hence, we must have, in $\mathbb{Z}_p$:

$$0 = {}^t\vec{d}\cdot\vec{s}\cdot y^2 \qquad \alpha\cdot y^2 + {}^t\vec{a}\cdot\vec{s}\cdot y^2 + a\cdot y = \delta\cdot y^2 + c\cdot y + {}^t\vec{c}\cdot\vec{s}\cdot y^3 \qquad \gamma_1\cdot y^2 = \epsilon_1\cdot y^2$$

Similarly, knowing all the s_i's, with only the SXSDL challenge $(G, y\cdot G, y^2\cdot G, \hat{G}, y\cdot\hat{G})$ for $y\neq 0$, if the relation is not trivial, with non-negligible probability, one can solve the quadratic equation and find y. Hence, we must have, in $\mathbb{Z}_p$:

$$0 = {}^t\vec{d}\cdot\vec{s} \qquad 0 = {}^t\vec{c}\cdot\vec{s} \qquad \alpha + {}^t\vec{a}\cdot\vec{s} = \delta \qquad a = c \qquad \gamma_1 = \epsilon_1$$

Eventually, keeping only one s_i unknown, if the relations are not trivial, with non-negligible probability, one can solve the linear equation and find s_i. Hence, by symmetry on i, we must have, in $\mathbb{Z}_p$:

$$\vec{d} = \vec{0} \qquad \vec{c} = \vec{0} \qquad \alpha = \delta \qquad \vec{a} = \vec{0} \qquad a = c \qquad \gamma_1 = \epsilon_1$$

In particular: $\gamma_1 = \epsilon_1$ and $\vec{d} = \sum_j \phi_{1,j}\cdot\vec{m}_{1,j} = \vec{0}$. By symmetry on k, we can extend to

$$\gamma_k = \epsilon_k \qquad \sum_j \phi_{k,j}\cdot\vec{m}_{k,j} = \vec{0} \qquad \text{for all } k$$

As a conclusion:

$$\hat{Q}^* = (\alpha\cdot xy + \gamma_1\cdot y + a\cdot x)\cdot\hat{G} \qquad Q^* = (\alpha\cdot xy + \gamma_1\cdot y + a\cdot x)\cdot G$$

A Unique List of Messages is Involved. In a second step, we show that exactly one tag (and the corresponding list of messages) is involved in the forgery. To this aim, we consider the validity of the signature:

$$\begin{aligned}e(\sigma^*,\hat{Q}^*) &= e\big(\big(\psi\cdot xy + \mu_1\cdot y + u\cdot x + {}^t\vec{u}\cdot\vec{s}\cdot xy^2 + {}^t\vec{v}\cdot\vec{s}\cdot x^2y\big)\cdot G,\\ &\qquad (\alpha\cdot xy + \gamma_1\cdot y + a\cdot x)\cdot\hat{G}\big)\\ &= \big(\psi\cdot\alpha\cdot x^2y^2 + \mu_1\cdot\alpha\cdot xy^2 + u\cdot\alpha\cdot x^2y + {}^t\vec{u}\cdot\vec{s}\cdot\alpha\cdot x^2y^3\\ &\qquad + {}^t\vec{v}\cdot\vec{s}\cdot\alpha\cdot x^3y^2 + \psi\cdot\gamma_1\cdot xy^2 + \mu_1\cdot\gamma_1\cdot y^2 + u\cdot\gamma_1\cdot xy\\ &\qquad + {}^t\vec{u}\cdot\vec{s}\cdot\gamma_1\cdot xy^3 + {}^t\vec{v}\cdot\vec{s}\cdot\gamma_1\cdot x^2y^2 + \psi\cdot a\cdot x^2y + \mu_1\cdot a\cdot xy\\ &\qquad + u\cdot a\cdot x^2 + {}^t\vec{u}\cdot\vec{s}\cdot a\cdot x^2y^2 + {}^t\vec{v}\cdot\vec{s}\cdot a\cdot x^3y\big)\cdot e(G,\hat{G})\\ = \prod_i e(M_i^*,\hat{P}_i) &= \prod_i e\big(\big(\rho_i\cdot xy + \zeta_{1,i}\cdot y + u'_i\cdot x + {}^t\vec{u}_i\cdot\vec{s}\cdot xy^2 + {}^t\vec{v}_i\cdot\vec{s}\cdot x^2y\big)\cdot G,\\ &\qquad s_i\cdot xy\cdot\hat{G}\big)\\ &= e\big(\sum_i \big(\rho_i\cdot s_i\cdot x^2y^2 + \zeta_{1,i}\cdot s_i\cdot xy^2 + u'_i\cdot s_i\cdot x^2y\\ &\qquad + {}^t\vec{u}_i\cdot\vec{s}\cdot s_i\cdot x^2y^3 + {}^t\vec{v}_i\cdot\vec{s}\cdot s_i\cdot x^3y^2\big)\cdot G,\hat{G}\big)\end{aligned}$$

Again, in basis $e(G,\hat{G})$, using the scalars x, y, and $\vec{s}$, this gives

$$\begin{aligned}&\psi\cdot\alpha\cdot x^2y^2 + \mu_1\cdot\alpha\cdot xy^2 + u\cdot\alpha\cdot x^2y + {}^t\vec{u}\cdot\vec{s}\cdot\alpha\cdot x^2y^3 + {}^t\vec{v}\cdot\vec{s}\cdot\alpha\cdot x^3y^2\\ &+ \psi\cdot\gamma_1\cdot xy^2 + \mu_1\cdot\gamma_1\cdot y^2 + u\cdot\gamma_1\cdot xy + {}^t\vec{u}\cdot\vec{s}\cdot\gamma_1\cdot xy^3 + {}^t\vec{v}\cdot\vec{s}\cdot\gamma_1\cdot x^2y^2\\ &+ \psi\cdot a\cdot x^2y + \mu_1\cdot a\cdot xy + u\cdot a\cdot x^2 + {}^t\vec{u}\cdot\vec{s}\cdot a\cdot x^2y^2 + {}^t\vec{v}\cdot\vec{s}\cdot a\cdot x^3y\\ &= \sum_i \rho_i\cdot s_i\cdot x^2y^2 + \zeta_{1,i}\cdot s_i\cdot xy^2 + u'_i\cdot s_i\cdot x^2y + {}^t\vec{u}_i\cdot\vec{s}\cdot s_i\cdot x^2y^3 + {}^t\vec{v}_i\cdot\vec{s}\cdot s_i\cdot x^3y^2\end{aligned}$$

Knowing all the s_i's, and y, with only the SXSDL challenge $(G, x \cdot G, x^2 \cdot G, \hat{G}, x \cdot \hat{G})$ for $x \neq 0$, if the relation is not trivial, with non-negligible probability, one can solve the quadratic equation and find x. Hence, we must have, in $\mathbb{Z}_p$:

$$\begin{aligned}
{}^t\vec{v} \cdot \vec{s} \cdot \alpha \cdot y^2 + {}^t\vec{v} \cdot \vec{s} \cdot a \cdot y &= \sum_i {}^t\vec{v}_i \cdot \vec{s} \cdot s_i \cdot y^2 \\
\psi \cdot \alpha \cdot y^2 + u \cdot \alpha \cdot y + {}^t\vec{u} \cdot \vec{s} \cdot \alpha \cdot y^3 + {}^t\vec{v} \cdot \vec{s} \cdot \gamma_1 \cdot y^2 + \psi \cdot a \cdot y + u \cdot a + {}^t\vec{u} \cdot \vec{s} \cdot a \cdot y^2 \\
&= \sum_i \rho_i \cdot s_i \cdot y^2 + u'_i \cdot s_i \cdot y + {}^t\vec{u}_i \cdot \vec{s} \cdot s_i \cdot y^3 \\
\mu_1 \cdot \alpha \cdot y^2 + \psi \cdot \gamma_1 \cdot y^2 + u \cdot \gamma_1 \cdot y + {}^t\vec{u} \cdot \vec{s} \cdot \gamma_1 \cdot y^3 + \mu_1 \cdot a \cdot y &= \sum_i \zeta_{1,i} \cdot s_i \cdot y^2 \\
\mu_1 \cdot \gamma_1 \cdot y^2 &= 0
\end{aligned}$$

Similarly, knowing all the s_i's, with only the SXSDL challenge $(G, y \cdot G, y^2 \cdot G, \hat{G}, y \cdot \hat{G})$ for $y \neq 0$, if the relation is not trivial, with non-negligible probability, one can solve the quadratic equation and find y. Hence, we must have, in $\mathbb{Z}_p$:

$$\begin{aligned}
{}^t\vec{v} \cdot \vec{s} \cdot \alpha &= \sum_i {}^t\vec{v}_i \cdot \vec{s} \cdot s_i & {}^t\vec{v} \cdot \vec{s} \cdot a &= 0 \\
{}^t\vec{u} \cdot \vec{s} \cdot \alpha &= \sum_i {}^t\vec{u}_i \cdot \vec{s} \cdot s_i & \psi \cdot \alpha + {}^t\vec{v} \cdot \vec{s} \cdot \gamma_1 + {}^t\vec{u} \cdot \vec{s} \cdot a &= \sum_i \rho_i \cdot s_i \\
u \cdot \alpha + \psi \cdot a &= \sum_i u'_i \cdot s_i & u \cdot a &= 0 \\
{}^t\vec{u} \cdot \vec{s} \cdot \gamma_1 &= 0 & \mu_1 \cdot \alpha + \psi \cdot \gamma_1 &= \sum_i \zeta_{1,i} \cdot s_i \\
u \cdot \gamma_1 + \mu_1 \cdot a &= 0 & \mu_1 \cdot \gamma_1 &= 0
\end{aligned}$$

Eventually, keeping only one s_i unknown, if the relations are not trivial, with non-negligible probability, one can solve the linear equation and find s_i. Hence, by symmetry on i, we must have, in $\mathbb{Z}_p$:

$$\begin{array}{ccccc}
\alpha \cdot \vec{v} = \vec{0} & \vec{v}_i = \vec{0}\ \forall i & a \cdot \vec{v} = \vec{0} & \alpha \cdot \vec{u} = \vec{0} & \vec{u}_i = \vec{0}\ \forall i \\
\psi \cdot \alpha = 0 & \gamma_1 \cdot \vec{v} + a \cdot \vec{u} = (\rho_i)_i & u \cdot \alpha + \psi \cdot a = 0 & (u'_i)_i = \vec{0} & u \cdot a = 0 \\
\gamma_1 \cdot \vec{u} = \vec{0} & \mu_1 \cdot \alpha + \psi \cdot \gamma_1 = 0 & (\zeta_{1,i})_i = \vec{0} & u \cdot \gamma_1 + \mu_1 \cdot a = 0 & \mu_1 \cdot \gamma_1 = 0
\end{array}$$

Which can be reordered as

$$\begin{array}{cccc}
\alpha \cdot \vec{u} = \vec{0} & \gamma_1 \cdot \vec{u} = \vec{0} & \alpha \cdot \vec{v} = \vec{0} & a \cdot \vec{v} = \vec{0} \\
\vec{u}_i = \vec{0}\ \forall i & \vec{v}_i = \vec{0}\ \forall i & & \\
\psi \cdot \alpha = 0 & u \cdot a = 0 & \mu_1 \cdot \gamma_1 = 0 & \\
\gamma_1 \cdot \vec{v} + a \cdot \vec{u} = (\rho_i)_i & (u'_i)_i = \vec{0} & (\zeta_{1,i})_i = \vec{0} & \\
u \cdot \alpha + \psi \cdot a = 0 & \mu_1 \cdot \alpha + \psi \cdot \gamma_1 = 0 & u \cdot \gamma_1 + \mu_1 \cdot a = 0 &
\end{array}$$

This simplifies into

$$\begin{aligned}
\vec{M}^* &= (\rho_i)_i \cdot xy \cdot G = (\gamma_1 \cdot \vec{v} + a \cdot \vec{u}) \cdot xy \cdot G \\
\sigma^* &= \left(\psi \cdot xy + \mu_1 \cdot y + u \cdot x + {}^t\vec{u} \cdot \vec{s} \cdot xy^2 + {}^t\vec{v} \cdot \vec{s} \cdot x^2 y\right) \cdot G
\end{aligned}$$

Hypothesis: $\alpha \neq 0 \bmod p$. This implies, in $\mathbb{Z}_p$:

$$\vec{u} = \vec{0} \qquad \vec{v} = \vec{0} \qquad \psi = 0 \qquad u = 0 \qquad \mu_1 = 0 \qquad (\rho_i)_i = 0$$

Then $\vec{M}^* = \vec{0}$ and $\sigma^* = 0$, which is refused as a valid pair. We thus have $\alpha = 0$, so $\psi \cdot a = \psi \cdot \gamma_1 = 0$.

Hypothesis: $a \neq 0 \bmod p$. This implies, in $\mathbb{Z}_p$:

$$\vec{v} = \vec{0} \qquad u = 0 \qquad \psi = 0 \qquad \mu_1 = 0 \qquad a \cdot \vec{u} = (\rho_i)_i$$

Then $\vec{u} \neq \vec{0}$, otherwise $\vec{M}^* = \vec{0}$, so $\gamma_1 = 0$:

$$\vec{M}^* = a \cdot \vec{u} \cdot xy \cdot G \qquad \sigma^* = {}^t\vec{u} \cdot \vec{s} \cdot xy^2 \cdot G \qquad Q^* = a \cdot x \cdot G$$

Hypothesis: $a = 0 \bmod p$. This implies $\gamma_1 \neq 0 \bmod p$, to avoid $\vec{M}^* = \vec{0}$, and then:

$$\vec{u} = \vec{0} \qquad \mu_1 = 0 \qquad \gamma_1 \cdot \vec{v} = (\rho_i)_i \qquad u = 0 \qquad \psi = 0$$

Then

$$\vec{M}^* = \gamma_1 \cdot \vec{v} \cdot xy \cdot G \qquad \sigma^* = {}^t\vec{v} \cdot \vec{s} \cdot x^2 y \cdot G \qquad Q^* = \gamma_1 \cdot y \cdot G$$

As a consequence, either

$$Q^* = a \cdot x \cdot G = \left(\sum_{k>1} \gamma_k/\tau'_k\right) \cdot x \cdot G = \left(\sum_{k>1} \gamma_k/\tau_k\right) \cdot P$$
$$\text{or } Q^* = \gamma_1 \cdot y \cdot G = \gamma_1/\tau_1 \cdot P$$

By symmetry, where either $\gamma_1 = 0$ or all the other $\gamma_k = 0$, for $k > 1$, then there must exist a unique k^* such that $\gamma_{k^*} \neq 0$:

$$Q^* = \gamma_{k^*}/\tau_{k^*} \cdot P \qquad \hat{Q}^* = \gamma_{k^*}/\tau_{k^*} \cdot \hat{P}$$

The Message is in the Vector-Subspace. Eventually, we show that the message $\vec{M}^*$ is in the appropriate vector-subspace. From the above notations, we can note that

$$\vec{M}^* = \gamma_{k^*} \cdot \left(\sum_j \nu_{k^*,j} \cdot \vec{m}_{k^*,j}\right) \cdot P = \gamma_{k^*} \cdot \sum_j \nu_{k^*,j} \cdot \vec{M}_{k^*,j}$$
$$\sigma^* = \tau_{k^*} \cdot \left(\sum_j \nu_{k^*,j} \cdot \vec{m}_{k^*,j}\right) \cdot \vec{s} \cdot P = \tau_{k^*} \cdot \sum_j \nu_{k^*,j} \cdot {}^t\vec{s} \cdot \vec{M}_{k^*,j}$$

Hence, for the unique index k^* such that $\gamma_{k^*} \neq 0$, with $\nu'_{k^*,j} = \gamma_{k^*} \cdot \nu_{k^*,j}$, for $j = 1, \ldots, K_{k^*}$,

$$\vec{M}^* = \sum_j \nu'_{k^*,j} \cdot \vec{M}_{k^*,j}$$

Remark. With overwhelming probability, we are in the above situation, with known coefficients $\nu'_{k^*,j}$, which leads to extractable unforgeability of the FHS signature, under selective message attacks, when keys and tags are honestly generated.

B Distributed Generation of the Global Parameters

As noted above, during the setup, one must generate, in a trusted way:
- the ballot encryption/decryption keys $(\mathsf{ek} = \vec{Z} = \vec{z} \cdot P, \mathsf{dk} = \vec{z})$
- the commitment verification/secret keys $(Y = y \cdot P, y)$
- the commitment key $\mathsf{ck} = (Q_0, \vec{Q}) \in \mathbb{G} \times \mathbb{G}^n$,
- the basic signatures $(\mathsf{VK}, (\Sigma_0, (\Sigma_i)_i, \Sigma_{n+1}), \mathsf{VK}', ((\mathsf{Tag}_{k,j}, \sigma_{k,j,0}, (\sigma_{k,j,i})_i, \sigma_{k,j,n+1})_j)_k)$

To this aim, we use the same type III pairing-friendly setting $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, p, P, \hat{P}, e)$ as global parameters param, according to the common security parameter κ. We assume all the users have a pair of encryption/decryption keys for receiving private informations.

B.1 Generation of the Keys

The Ballot Encryption/Decryption Keys. To generate dk and ek in a distributed way, between N clients with a threshold T to reconstruct, the i-th client

- generates n random polynomials $(P_{i,k})_k$ of degree $T-1$ in $\mathbb{Z}_p$;
- publishes $U_{i,k,j} = P_{i,k}(j) \cdot P \in \mathbb{G}$, for $j = 0, 1, \ldots, N$;
- sends $u_{i,k,j} = P_{i,k}(j) \in \mathbb{Z}_p$ to the j-th client, in a private way.

Everybody can compute $Z_k = \sum_i U_{i,k,0} = \sum_i P_{i,k}(0) \cdot P = P_k(0) \cdot P$, for the polynomials $P_k = \sum_i P_{i,k}$ of degree $T-1$. This constitutes the encryption key $\mathsf{ek} = \vec{Z} = (Z_k)_k$.

Individually, each j-th client can compute $z_{j,k} = \sum_i u_{i,k,j} = \sum_i P_{i,k}(j) = P_k(j)$. Then, any subset S of T clients can use Lagrange interpolation to build $z_k = \sum_{j \in S} \lambda_j \cdot z_{j,k}$ such that $Z_k = z_k \cdot P$. Hence, $\vec{z} = \sum_{j \in S} \lambda_j \cdot \vec{z}_j$ such that $\vec{Z} = \vec{z} \cdot P$.

This is enough to decrypt the ciphertext $(F_0, \vec{F})$ without reconstructing the decryption key:

- each j-th client in S computes and publishes $\vec{V}_j = \vec{z}_j \cdot F_0$;
- everybody can compute $\vec{V} = \sum_{j \in S} \vec{V}_j = \sum_{j \in S} \vec{z}_j \cdot F_0 = \vec{z} \cdot F_0$;
- everybody can compute $\vec{m} \in \mathbb{Z}_p^n$ such that $\vec{F} - \vec{V} = \vec{m} \cdot P$.

This protocol is secure against honest-but-curious clients. To prevent malicious behaviors, all the publication rounds can be first committed. The validity of the $(U_{i,k,j})_{i,k,j}$ can be checked with linear combinations; and the validity of $\vec{V}_j$ can be proven with a zero-knowledge proof *à la Schnorr*.

The Commitment Verification/Secret Keys. The same approach as above can be used to generate a secret sharing of $y \in \mathbb{Z}_p$, with $Y = y \cdot P$, between N clients with a threshold T to reconstruct.

The Commitment Key. Using a hash function $\mathcal{H}$ onto $\mathbb{G}$, one can publicly define

$$Q_0 = \mathcal{H}(\mathsf{Election}, \text{“Commitment”}, 0), \quad Q_i = \mathcal{H}(\mathsf{Election}, \text{“Commitment”}, i), \text{ for } i = 1, \ldots, n,$$

where $\mathsf{Election}$ is a bit-string that describes the current election.

B.2 Generation of the Linearly-Homomorphic Signatures

In addition to be linearly-homomorphic on the messages, our signatures are linearly-homomorphic on the keys:

- if $\mathsf{sk} = \sum_j \lambda_j \cdot \mathsf{sk}_j$ and $\sigma_j = \mathsf{LH\text{-}Sign.Sign}(\mathsf{sk}_j, \vec{m})$, for the same message $\vec{m}$, then $\sum_i \lambda_j \cdot \sigma_j = \mathsf{LH\text{-}Sign.Sign}(\mathsf{sk}, \vec{m})$;
- Similarly, if $\mathsf{sk} = \sum_j \lambda_j \cdot \mathsf{sk}_j$ and $\sigma_j = \mathsf{LH\text{-}Sign\text{-}RTag.Sign}(\mathsf{sk}_j, \tau, \vec{m})$, for the same secret tag τ and the same message $\vec{m}$, then $\sum_i \lambda_j \cdot \sigma_j = \mathsf{LH\text{-}Sign\text{-}RTag.Sign}(\mathsf{sk}, \tau, \vec{m})$.

The former property allows to generate VK and $(\mathsf{SK}_j)_j$, and the signatures $(\Sigma_0, (\Sigma_i)_i, \Sigma_{n+1})$, as above. However, for the $\mathsf{LH\text{-}Sign\text{-}RTag}$, this is a bit more complex, so focus to the 2-party case, with $\mathcal{P}_1$ and $\mathcal{P}_2$ that do not collude. The security analysis is provided in the semi-honest setting.

Description. We start with the first kind of signatures, for all the messages $(\vec{M}_i)_i$ in parallel:

- Client $\mathcal{P}_1$
  - chooses $\mathsf{SK}_1 \xleftarrow{\$} \mathbb{Z}_p^{n+3}$;
  - computes $\mathsf{VK}_1 = \mathsf{SK}_1 \cdot \hat{P}$;
  - generates $(\Sigma_{1,i} = {}^t\mathsf{SK}_1 \cdot \vec{M}_i)_i$;
  - deletes SK_1;
  - publishes VK_1, $(\Sigma_{1,i})_i$;
- Client $\mathcal{P}_2$
  - chooses $\mathsf{SK}_2 \xleftarrow{\$} \mathbb{Z}_p^{n+3}$;
  - computes $\mathsf{VK}_2 = \mathsf{SK}_2 \cdot \hat{P}$;
  - generates $(\Sigma_{2,i} = {}^t\mathsf{SK}_2 \cdot \vec{M}_i)_i$;

  - deletes SK_2;
  - publishes VK_2, $(\Sigma_{2,i})_i$;
- Everybody
  - computes $\mathsf{VK} = \mathsf{VK}_1 + \mathsf{VK}_2$;
  - computes $(\Sigma_i = \Sigma_{1,i} + \Sigma_{2,i})_i$;
  - checks $\mathsf{LH\text{-}Sign.Verif}(\mathsf{VK}, \vec{M}_i, \Sigma_i)$, for all i.

One can note that $\mathsf{VK} = \mathsf{VK}_1 + \mathsf{VK}_2 = (\mathsf{SK}_1 + \mathsf{SK}_2) \cdot \hat{P} = \mathsf{SK} \cdot \hat{P}$, for a virtual signing key $\mathsf{SK} = \mathsf{SK}_1 + \mathsf{SK}_2$, and for $i = 0, \ldots, n+1$, $\Sigma_i = \Sigma_{1,i} + \Sigma_{2,i} = {}^t(\mathsf{SK}_1 + \mathsf{SK}_2) \cdot \vec{M}_i = {}^t\mathsf{SK} \cdot \vec{M}_i = \mathsf{LH\text{-}Sign.Sign}(\mathsf{SK}, \vec{M}_i)$. In addition, we have, for $i = 0, \ldots, n+1$, $\Sigma_{1,i} = \mathsf{LH\text{-}Sign.Sign}(\mathsf{SK}_1, \vec{M}_i)$ and $\Sigma_{2,i} = \mathsf{LH\text{-}Sign.Sign}(\mathsf{SK}_2, \vec{M}_i)$. They can be verified.

Security in the Semi-Honest Setting. The view of the players is

$$\begin{array}{ll} \mathsf{VK}_1 = \mathsf{SK}_1 \cdot \hat{P} & (\Sigma_{1,i} = {}^t\mathsf{SK}_1 \cdot \vec{M}_i)_i \\ \mathsf{VK}_2 = \mathsf{SK}_2 \cdot \hat{P} & (\Sigma_{2,i} = {}^t\mathsf{SK}_2 \cdot \vec{M}_i)_i \end{array}$$

For $\mathcal{P}_1$ who additionally knows SK_1, this is equivalent to the following distributions, for random scalars $\mathsf{SK}_1, \mathsf{SK}_2 \overset{\$}{\leftarrow} \mathbb{Z}_p^{n+3}$,

$$\begin{array}{rlllll} \mathcal{D} = \{\mathsf{SK}_1 & \mathsf{VK}_1 = \mathsf{SK}_1 \cdot \hat{P} & \mathsf{VK}_2 = \mathsf{SK}_2 \cdot \hat{P} & (\Sigma_{1,i} = {}^t\mathsf{SK}_1 \cdot \vec{M}_i & \Sigma_{2,i} = {}^t\mathsf{SK}_2 \cdot \vec{M}_i)_i\} \\ = \{\mathsf{SK}_1 & \mathsf{VK}_1 = \mathsf{SK}_1 \cdot \hat{P} & \mathsf{VK} - \mathsf{VK}_1 & (\Sigma_{1,i} = {}^t\mathsf{SK}_1 \cdot \vec{M}_i & \Sigma_{2,i} = \Sigma_i - {}^t\mathsf{SK}_1 \cdot \vec{M}_i)_i\} \\ = \{\mathsf{SK}_1 & \mathsf{VK}_1 = \mathsf{SK}_1 \cdot \hat{P} & \mathsf{VK} & (\Sigma_{1,i} = {}^t\mathsf{SK}_1 \cdot \vec{M}_i & \Sigma_{2,i} = \Sigma_i)_i\} \end{array}$$

for random scalars $\mathsf{SK}_1 \overset{\$}{\leftarrow} \mathbb{Z}_p^{n+3}$, and signatures $(\Sigma_i)_i$ of $(\vec{M}_i)_i$ under a random verification key $\mathsf{VK} \overset{\$}{\leftarrow} \hat{\mathbb{G}}^{n+3}$. Because of the symmetry, the same analysis holds for $\mathcal{P}_2$. This proves that no additional information leaks beyond the signatures on the fixed messages under a random key VK, to a semi-honest adversary (honest-but-curious). Which was the information provided by the trusted setup.

B.3 Generation of the Linearly-Homomorphic Signatures with Randomizable Tags.

Again, we focus to the 2-party case, with $\mathcal{P}_1$ and $\mathcal{P}_2$ that do not collude. The security analysis is provided in the semi-honest setting.

Description. All the tags, indexed by t, and all the messages $\vec{M}_{t,i}$, are dealt in parallel, with $R_{t,i} = \mathcal{H}(\mathsf{Election}, \text{“Signatures”}, t, i) \in \mathbb{G}$:
- Client $\mathcal{P}_1$
  - chooses $\mathsf{SK}'_1 \overset{\$}{\leftarrow} \mathbb{Z}_p^3$;
  - computes $\mathsf{VK}'_1 = \mathsf{SK}'_1 \cdot \hat{P}$;
  - chooses $(\alpha_{t,1} \overset{\$}{\leftarrow} \mathbb{Z}_p)_t$;
  - computes $(\sigma_{t,i,1} = \alpha_{t,1} \cdot ({}^t\mathsf{SK}'_1 \cdot \vec{M}_{t,i} + R_{t,i}))_{t,i}$;
  - sends VK'_1 and $(\sigma_{t,i,1})_{t,i}$ to $\mathcal{P}_2$;
- Client $\mathcal{P}_2$
  - chooses $\mathsf{SK}'_2 \overset{\$}{\leftarrow} \mathbb{Z}_p^3$;
  - computes $\mathsf{VK}'_2 = \mathsf{SK}'_2 \cdot \hat{P}$;
  - chooses $(\alpha_{t,2} \overset{\$}{\leftarrow} \mathbb{Z}_p)_t$;
  - computes $(\sigma_{t,i,2} = \alpha_{t,2} \cdot ({}^t\mathsf{SK}'_2 \cdot \vec{M}_{t,i} - R_{t,i}))_{t,i}$ and $(\sigma'_{t,i,1} = \alpha_{t,2} \cdot \sigma_{t,i,1})_{t,i}$;

$$\sigma'_{t,i,1} = \alpha_{t,1}\alpha_{t,2} \cdot ({}^t\mathsf{SK}'_1 \cdot \vec{M}_{t,i} + R_{t,i})$$

  - sends VK'_2 and $(\sigma_{t,i,2}, \sigma'_{t,i,1})_{t,i}$ to $\mathcal{P}_1$;
- Client $\mathcal{P}_1$
  - chooses $(\tau_{t,1} \overset{\$}{\leftarrow} \mathbb{Z}_p)_t$;

  - computes $(Q_{t,1} = 1/\tau_{t,1} \cdot P, \hat{Q}_{t,1} = 1/\tau_{t,1} \cdot \hat{P})_t$;
  - computes $(\sigma'_{t,i} = \tau_{t,1} \cdot (\sigma_{t,i,2} + 1/\alpha_{t,1} \cdot \sigma'_{t,i,1}))_{t,i}$;

$$\sigma'_{t,i} = \tau_{t,1}\alpha_{t,2} \cdot ({}^t\mathsf{SK}'_2 \cdot \vec{M}_{t,i} - R_{t,i}) + \tau_{t,1}\alpha_{t,2} \cdot ({}^t\mathsf{SK}'_1 \cdot \vec{M}_{t,i} + R_{t,i}) = \tau_{t,1}\alpha_{t,2} \cdot {}^t(\mathsf{SK}'_1 + \mathsf{SK}'_2) \cdot \vec{M}_{t,i}$$

  - sends $(\sigma'_{t,i})_{t,i}$ and $(Q_{t,1}, \hat{Q}_{t,1})_t$ to $\mathcal{P}_2$;
- Client $\mathcal{P}_2$
  - chooses $(\tau_{t,2} \overset{\$}{\leftarrow} \mathbb{Z}_p)_t$;
  - computes $(Q_t = 1/\tau_{t,2} \cdot Q_{t,1}, \hat{Q}_t = 1/\tau_{t,2} \cdot \hat{Q}_{t,1})_t$;

$$(Q_t, \hat{Q}_t) = (1/\tau_{t,1}\tau_{t,2} \cdot P, 1/\tau_{t,1}\tau_{t,2} \cdot \hat{P}) = (1/\tau_t \cdot P, 1/\tau_t \cdot \hat{P})$$

  - computes $(\sigma_{t,i} = \tau_{t,2}/\alpha_{t,2} \cdot \sigma'_{t,i})_{t,i}$;

$$\sigma_{t,i} = \sigma'_{t,i} = \tau_{t,1}\tau_{t,2} \cdot {}^t(\mathsf{SK}'_1 + \mathsf{SK}'_2) \cdot \vec{M}_{t,i}$$

  - sends $(\sigma_{t,i})_{t,i}$ and $(Q_t, \hat{Q}_t)_t$ to $\mathcal{P}_1$;
- Everybody
  - computes $\mathsf{VK}' = \mathsf{VK}'_1 + \mathsf{VK}'_2$;
  - stores $(Q_t, \hat{Q}_t)_t$ and $(\sigma_{t,i})_{t,i}$;
  - checks $\mathsf{LH\text{-}Sign\text{-}RTag.Verif}(\mathsf{VK}', (Q_t, \hat{Q}_t), \vec{M}_{t,i}, \sigma_{t,i})$, for all t and i.

Security in the Semi-Honest Setting. The view of the players is

$$\begin{array}{lll}
\mathsf{VK}'_1 = \mathsf{SK}'_1 \cdot \hat{P} & (\sigma_{t,i,1} = \alpha_{t,1} \cdot ({}^t\mathsf{SK}'_1 \cdot \vec{M}_{t,i} + R_{t,i}))_{t,i} & \\
\mathsf{VK}'_2 = \mathsf{SK}'_2 \cdot \hat{P} & (\sigma'_{t,i,1} = \alpha_{t,1}\alpha_{t,2} \cdot ({}^t\mathsf{SK}'_1 \cdot \vec{M}_{t,i} + R_{t,i}) & \sigma_{t,i,2} = \alpha_{t,2} \cdot ({}^t\mathsf{SK}'_2 \cdot \vec{M}_{t,i} - R_{t,i}))_{t,i} \\
& (\sigma'_{t,i} = \tau_{t,1}\alpha_{t,2} \cdot {}^t(\mathsf{SK}'_1 + \mathsf{SK}'_2) \cdot \vec{M}_{t,i})_{t,i} & (Q_{t,1} = 1/\tau_{t,1} \cdot P, \hat{Q}_{t,1} = 1/\tau_{t,1} \cdot \hat{P})_t \\
& (\sigma_{t,i} = \tau_{t,1}\tau_{t,2} \cdot {}^t(\mathsf{SK}'_1 + \mathsf{SK}'_2) \cdot \vec{M}_{t,i})_{t,i} & (Q_t = 1/\tau_{t,1}\tau_{t,2} \cdot P, \hat{Q}_t = 1/\tau_{t,1}\tau_{t,2} \cdot \hat{P})_t
\end{array}$$

For $\mathcal{P}_1$ who knows SK'_1 and $(\alpha_{t,1}, \tau_{t,1})_t$, this is equivalent to,

$$\begin{array}{llll}
\mathsf{SK}'_1 & \mathsf{SK}'_2 \cdot \hat{P} & (\tau_{t,1})_t & (1/\tau_{t,1}\tau_{t,2} \cdot P, 1/\tau_{t,1}\tau_{t,2} \cdot \hat{P})_t \\
& & (\alpha_{t,2} \cdot ({}^t\mathsf{SK}'_1 \cdot \vec{M}_{t,i} + R_{t,i}) & \alpha_{t,2} \cdot ({}^t\mathsf{SK}'_2 \cdot \vec{M}_{t,i} - R_{t,i}))_{t,i} \\
& & (\alpha_{t,2} \cdot {}^t(\mathsf{SK}'_1 + \mathsf{SK}'_2) \cdot \vec{M}_{t,i} & \tau_{t,1}\tau_{t,2} \cdot {}^t(\mathsf{SK}'_1 + \mathsf{SK}'_2) \cdot \vec{M}_{t,i})_{t,i}
\end{array}$$

for random scalars $\mathsf{SK}'_1, \mathsf{SK}'_2 \overset{\$}{\leftarrow} \mathbb{Z}_p^3$, and $(\alpha_{t,1}, \tau_{t,1}, \alpha_{t,2}, \tau_{t,2} \overset{\$}{\leftarrow} \mathbb{Z}_p)_t$. This is thus equivalent to the following view

$$\begin{array}{lllll}
\mathsf{SK}'_1 & \mathsf{SK}' \cdot \hat{P} & (\tau_{t,1})_t & (1/\tau_t \cdot P, 1/\tau_t \cdot \hat{P})_t & \\
& & (\alpha_{t,2} \cdot {}^t\mathsf{SK}'_1 \cdot \vec{M}_{t,i} + \alpha_{t,2} \cdot R_{t,i} & \alpha_{t,2} \cdot {}^t\mathsf{SK}' \cdot \vec{M}_{t,i} & \tau_t \cdot {}^t\mathsf{SK}' \cdot \vec{M}_{t,i})_{t,i}
\end{array}$$

for random scalars $\mathsf{SK}'_1, \mathsf{SK}' \overset{\$}{\leftarrow} \mathbb{Z}_p^3$, and $(\tau_{t,1}, \tau_t, \alpha_{t,2} \overset{\$}{\leftarrow} \mathbb{Z}_p)_t$. Even if one would know $\vec{m}_{t,i} \in \mathbb{Z}_p^3$ such that $\vec{M}_{t,i} = \vec{m}_{t,i} \cdot P$, this is

$$\begin{array}{lllll}
\mathsf{SK}'_1 & \mathsf{SK}' \cdot \hat{P} & (\tau_{t,1})_t & (1/\tau_t \cdot P, 1/\tau_t \cdot \hat{P})_t & \\
& & ({}^t\mathsf{SK}'_1 \cdot \vec{m}_{t,i} \cdot \alpha_{t,2} \cdot P + \alpha_{t,2} \cdot R_{t,i} & {}^t\mathsf{SK}' \cdot \vec{m}_{t,i} \cdot \alpha_{t,2} \cdot P & {}^t\mathsf{SK}' \cdot \vec{m}_{t,i} \cdot \tau_t \cdot P
\end{array}$$

And under the DDH assumption in $\mathbb{G}$,

$$(P, R_{t,i}, \alpha_{t,2} \cdot P, \alpha_{t,2} \cdot R_{t,i}) \approx (P, R_{t,i}, \alpha_{t,2} \cdot P, R'_{t,i}).$$

Hence, the above view is indistinguishable from

$$\begin{array}{lllll}
\mathsf{SK}'_1 & \mathsf{SK}' \cdot \hat{P} & (\tau_{t,1})_t & (1/\tau_t \cdot P, 1/\tau_t \cdot \hat{P})_t & \\
& & (\alpha_{t,2} \cdot {}^t\mathsf{SK}'_1 \cdot \vec{M}_{t,i} + R'_{t,i} & \alpha_{t,2} \cdot {}^t\mathsf{SK}' \cdot \vec{M}_{t,i} & \tau_t \cdot {}^t\mathsf{SK}' \cdot \vec{M}_{t,i})_{t,i}
\end{array}$$

for random scalars $\mathsf{SK}'_1, \mathsf{SK}' \xleftarrow{\$} \mathbb{Z}_p^3$, and $(\tau_{t,1}, \tau_t, \alpha_{t,2} \xleftarrow{\$} \mathbb{Z}_p)_t$, and random $(R'_{t,i} \xleftarrow{\$} \mathbb{G})_{t,i}$. Which is indistinguishable from

$$\begin{array}{lllll}
\mathsf{SK}'_1 & \mathsf{SK}' \cdot \hat{P} & (\tau_{t,1})_t & (1/\tau_t \cdot P, 1/\tau_t \cdot \hat{P})_t & \\
 & & (A_{t,i} & \alpha_{t,2} \cdot {}^t\mathsf{SK}' \cdot \vec{M}_{t,i} & \tau_t \cdot {}^t\mathsf{SK}' \cdot \vec{M}_{t,i})_{t,i}
\end{array}$$

for random scalars $\mathsf{SK}'_1, \mathsf{SK}' \xleftarrow{\$} \mathbb{Z}_p^3$, and $(\tau_{t,1}, \tau_t, \alpha_{t,2} \xleftarrow{\$} \mathbb{Z}_p)_t$, and random $(A_{t,i} \xleftarrow{\$} \mathbb{G})_{t,i}$, and thus from

$$\begin{array}{lllll}
\mathsf{SK}'_1 & \mathsf{VK}' & (\tau_{t,1})_t & (Q_t, \hat{Q}_t)_t & \\
 & & (A_{t,i} & B_{t,i} & \sigma_{t,i})_{t,i}
\end{array}$$

for random scalars $\mathsf{SK}'_1 \xleftarrow{\$} \mathbb{Z}_p^3$, and $(\tau_{t,1} \xleftarrow{\$} \mathbb{Z}_p)_t$ and random group elements $(A_{t,i}, B_{t,i} \xleftarrow{\$} \mathbb{G})_{t,i}$, and signatures with randomizable tags $(Q_t, \hat{Q}_t)_t$, $(\sigma_{t,i})_{t,i}$ of $(\vec{M}_{t,i})_{t,i}$ under a random verification key VK'.

For $\mathcal{P}_2$ who knows SK'_2 and $(\tau_{t,2})_t$, this is equivalent to,

$$\begin{array}{llll}
\mathsf{SK}'_2 & \mathsf{SK}'_1 \cdot \hat{P} & (\tau_{t,2})_t & (1/\tau_{t,1}\tau_{t,2} \cdot P, 1/\tau_{t,1}\tau_{t,2} \cdot \hat{P})_t \\
 & & (\alpha_{t,1} \cdot ({}^t\mathsf{SK}'_1 \cdot \vec{M}_{t,i} + R_{t,i}) & \tau_{t,1}\tau_{t,2} \cdot {}^t(\mathsf{SK}'_1 + \mathsf{SK}'_2) \cdot \vec{M}_{t,i})_{t,i}
\end{array}$$

for random scalars $\mathsf{SK}'_1, \mathsf{SK}'_2 \xleftarrow{\$} \mathbb{Z}_p^3$, and $(\tau_{t,1}, \tau_{t,2}, \alpha_{t,1} \xleftarrow{\$} \mathbb{Z}_p)_t$, which view is indistinguishable, under the DDH assumption in $\mathbb{G}$, from

$$\begin{array}{llll}
\mathsf{SK}'_2 & \mathsf{SK}'_1 \cdot \hat{P} & (\tau_{t,2})_t & (1/\tau_t \cdot P, 1/\tau_t \cdot \hat{P})_t \\
 & & (\alpha_{t,1} \cdot {}^t\mathsf{SK}'_1 \cdot \vec{M}_{t,i} + R'_{t,i} & \tau_t \cdot {}^t(\mathsf{SK}'_1 + \mathsf{SK}'_2) \cdot \vec{M}_{t,i})_{t,i}
\end{array}$$

for random scalars $\mathsf{SK}'_1, \mathsf{SK}'_2 \xleftarrow{\$} \mathbb{Z}_p^3$, and $(\tau_t, \tau_{t,2}, \alpha_{t,1} \xleftarrow{\$} \mathbb{Z}_p)_t$ and random group elements $(R'_{t,i} \xleftarrow{\$} \mathbb{G})_{t,i}$. The above view is indistinguishable from

$$\begin{array}{llll}
\mathsf{SK}'_2 & \mathsf{VK}' & (\tau_{t,2})_t & (Q_t, \hat{Q}_t)_t \\
 & & (A_{t,i} & \sigma_{t,i})_{t,i}
\end{array}$$

for random scalars $\mathsf{SK}'_2 \xleftarrow{\$} \mathbb{Z}_p^3$, and $(\tau_{t,2} \xleftarrow{\$} \mathbb{Z}_p)_t$ and random group elements $(A_{t,i} \xleftarrow{\$} \mathbb{G})_{t,i}$, and signatures with randomizable tags $(Q_t, \hat{Q}_t)_t$, $(\sigma_{t,i})_{t,i}$ of $(\vec{M}_{t,i})_{t,i}$ under a random verification key VK'.

This proves that no additional information leaks beyond the signatures on the fixed messages with random tags under a random verification key VK', to a semi-honest adversary (honest-but-curious), unless one can break the DDH assumption in $\mathbb{G}$. We stress that this setup is important for the soundness of the validity proofs, but it does not impact the privacy property, that is unconditional.