Linearly-Homomorphic Signatures and Scalable Mix-Nets

Chloé Hébant[1,2], Duong Hieu Phan[3], and David Pointcheval[1,2]

[1] DIENS, École normale supérieure, CNRS, PSL University, Paris, France
[2] INRIA, Paris, France
[3] Université de Limoges, France

Abstract Anonymity is a primary ingredient for our digital life. Several tools have been designed to address it such as, for authentication, blind signatures, group signatures or anonymous credentials and, for confidentiality, randomizable encryption or mix-nets. When it comes to complex electronic voting schemes, random shuffling of ciphertexts with mix-nets is the only known tool. However, it requires huge and complex zero-knowledge proofs to guarantee the actual permutation of the initial ciphertexts.
In this paper, we propose a new approach for proving correct shuffling: the mix-servers can simply randomize individual ballots, which means the ciphertexts, the signatures, and the verification keys, with an additional global proof of constant size, and the output will be publicly verifiable. The computational complexity for the mix-servers is linear in the number of ciphertexts. Verification is also linear in the number of ciphertexts, independently of the number of rounds of mixing. This leads to the most efficient technique, that is highly scalable.Our constructions make use of linearly-homomorphic signatures, with new features, that are of independent interest.

Keywords: Anonymity, random shuffling, linearly-homomorphic signatures

1 Introduction

A shuffle of ciphertexts is a set of ciphertexts of the same plaintexts but in a permuted order such that it is not possible to trace back the senders after decryption. It can be used as a building block to anonymously send messages: if several servers perform a shuffle successively, nobody can trace the messages. More precisely, one honest mix-server suffices to mask the order of the ciphertexts and this produces a random permuted set of messages. These mix-servers constitute the notion of a mix-net protocol introduced by Chaum [Cha81]. The higher the number of mix-servers is, the safer the protocol gets, as one honest mix-server is enough even if all the other ones are dishonest. But the more costly the protocol becomes!

1.1 State of the Art

Usually, a shuffle of ciphertexts is a permutation applied to randomized ciphertexts. Randomization of the ciphertexts provides the privacy guarantee, but one additionally needs to prove the permutation property. This last step requires huge and complex zero-knowledge proofs. In the main two techniques, Furukawa and Sako [FS01] make proofs of permutation matrices and Neff [Nef01] considers polynomials which remain identical with a permutation of the roots. While the latter approach produces the most efficient schemes, they need to be interactive: more recently, Groth and Ishai [GI08] exploited this interactive approach and proposed a zero-knowledge argument for the correctness of a shuffle with sub-linear communication complexity, but computational complexity is super-linear. Since this is an interactive Special Honest-Verifier Zero-Knowledge protocol with public random coins, the Fiat-Shamir heuristic [FS87] can be applied to make it non-interactive in the random oracle model. However, with multiple mixing steps, which are required if one wants to guarantee anonymity even if some mix-servers are malicious, the communication is linear in this number of steps, and the verification cost becomes prohibitive.

The former approach with proof of permutation matrix is more classical, with many candidates. Groth and Lu [GL07] proposed the first non-interactive zero-knowledge proof of shuffle without random oracles, using Groth-Sahai proofs with pairings [GS08], but under non-standard computational assumptions that hold in the bilinear generic group model. Anyway, computations are still very high, the overhead proof is linear in Nn, where n is the number of ballots and N the number of shuffles, which is a bottleneck. In addition, they needed a Common Reference String (CRS) linear in n.

We propose a totally new approach that can handle each ballot in an independent way, with just a constant-size overhead. In addition, the overhead after each shuffle can be updated, to still keep it constant-size, independently of the number of mixing steps. From our knowledge, this is the most efficient and scalable solution: it will also rely on Groth-Sahai proofs with pairings [GS08] and under a new computational assumption that holds in the bilinear generic group model. As a consequence, assumptions are quite similar to [GL07], but we will just have constant-size CRS and constant-size overhead proof.

1.2 Our Approach

In a mix-net, each ciphertext C_i (encrypted vote in the ballot, in the context of electronic voting) is signed by its sender and the mix-server randomizes the ciphertexts $\{C_i\}$ and permutes them into the set $\{C'_i\}$ in a provable way. The goal of the proof is to show the existence of a permutation Π such that for every C_i in the input ballot-box, there is a randomization $C'_{\Pi(i)}$ in the output ballot-box, and vice-versa. Then, the output ciphertexts can be mixed again by another mix-server, hence the notion of mix-network (a.k.a. mix-net).

Our approach avoids the proof of an explicit permutation on all the ciphertexts (per mixing step) but still guarantees the appropriate properties deeply using the linearly-homomorphic signature schemes:

- each voter is associated to a signing/verification key-pair for a linearly-homomorphic signature scheme [BFKW09], and uses it to sign his ciphertext and a way to randomize it. This guarantees that the mix-server will only be able to generate new signatures on randomized ciphertexts, which are unlinkable to the original ciphertexts, due to the new random coins. However, the verification keys and signatures still allow linkability;
- each verification key of the voters is also signed with a linearly-homomorphic signature scheme, that allows randomization too, and provides unlinkability.

When talking about linearly-homomorphic signature schemes, we consider signatures that are malleable and that allow to sign any linear combination of the already signed vectors [BFKW09], this is thus an homomorphism on the messages. In order to be able to use this property on the latter scheme that signs the verification keys of the former scheme, it will additionally require some homomorphism on the keys.

However, whereas ciphertexts are signed under different keys, which excludes combinations, the verification keys are all signed under the authority's key. And a linearly-homomorphic signature scheme not only allows multiplication by a constant, but also linear combinations, which would allow combinations of keys and thus, possibly, of ballots. In order to avoid such combinations, we require the additional notion of *non-miscibility*: only linear combination on *one* message is possible, and thus just the multiplication by a constant.

We will thus use two distinct notations: $(\mathsf{Keygen}, \mathsf{Sign}, \mathsf{Verif})$ for a linearly-homomorphic signature, and $(\mathsf{Keygen}^*, \mathsf{Sign}^*, \mathsf{Verif}^*)$ for a linearly-homomorphic signature that additionally guarantees non-miscibility. We first provide a generic technique that can apply to any linearly-homomorphic signature: by adding Square Diffie-Hellman tuples $(g, g^{w_i}, g^{w_i^2})$ in the signed vector (later called expanded vector), we get non-miscibility. This indeed limits signatures to multiplications by constant values only. We then present and exploit a dedicated scheme, that is more efficient. These constructions present different advantages and constraints: the former is more

generic, while the latter is more efficient. We will thus focus on the latter in the body of the paper, and give details on the former in the appendices.

Unforgeability of the signature schemes together with the non-miscibility property will essentially provide the soundness of the proof of correct mixing (and thus for both constructions): only permutations of ballots are possible. Eventually, unlinkability (a.k.a zero-knowledge property) will be satisfied thanks to the randomizations that are indistinguishable for various users, under some DDH-like assumptions. With the above linear homomorphisms of the signatures, we can indeed guarantee that the output C'_j is a randomization of an input C_i, and the verification keys are unlinkable. But the proofs must be specific to the actual signature schemes.

More precisely, the signature unforgeability will guarantee that all the ballots in the output ballot-box come from legitimate signers: we will also have to make sure that there is no duplicates, nor new ballots, and the same numbers of ballots in the input ballot-box and output ballot-box for the formal proof of permutation.

This technique of randomizing ciphertexts and verification keys, and adapting signatures, can be seen as an extension of signatures on randomizable ciphertexts [BFPV11] which however did not allow updates of the verification keys. This previous approach excluded anonymity because of the invariant verification keys. Our new approach can find more applications where anonymity and privacy are crucial properties.

1.3 Related Work

Groth and Lu [GL07] protocol is definitely the closest approach to ours. This was the first efficient non-interactive zero-knowledge proof for correctness of a shuffle, in the standard model. They use BBS encryptions [BBS04] to have only 3 group elements in a ciphertext and Groth-Sahai proofs [GS08] to obtain a zero-knowledge proof of the implicit permutation matrix. As us, they rely on a non-standard assumption that holds in the generic bilinear group model. However, their assumption is very close to their goal, as it is a *permutation pairing assumption* that essentially assumes there exists a permutation when manipulating Square Diffie-Hellman tuples with some invariant requirements. They also required a quite huge CRS. In our case, for the general conversion that provides non-miscibility, we assume the existence of an extractor for linear combinations of signatures, which is close to the extractability assumptions used in SNARKs [GW11, BCCT12, GGPR13]. All our properties are proven in the generic bilinear group model, and we get a construction with a constant-size proof.

Indeed, the main drawback of their paper is the size of the proof: to shuffle n ciphertexts, the proof consists of $15n + 120$ group elements and because one needs to commit one scalar and $2n + 10$ group elements and to satisfy n quadratic equations and 6 equations of n-element products, the prover needs to make at least $3 \times 1 + 3 \times (2n + 10) + 6 \times (6n + 6) = 48n + 33$ exponentiations in a symmetric bilinear group. Whereas with our construction, the prover only needs to make 8 exponentiations in $\mathbb{G}_1$ and 9 exponentiations in $\mathbb{G}_2$ per ciphertext plus a constant overhead of 7 exponentiations in $\mathbb{G}_1$ and 5 exponentiations in $\mathbb{G}_2$. In addition, from the communication point of view, after N mixing protocols, our global proof does not grow and just consists of 4 group elements, whatever the value N.

1.4 Organization

In the next section, we recall some usual assumptions in pairing-based groups, and we introduce a new *unlinkability assumption* that will be one of the core assumptions of our applications. Note that it holds in the generic bilinear group model. In Section 3, we recall the notion of linearly-homomorphic signatures, with a compact construction and its security analysis in the generic bilinear group model. Then we extend it to handle the non-miscibility property. We then apply these constructions to mix-networks (Section 4), followed by a detailed security analysis in Section 5. Eventually, we explain application to electronic voting in Section 6.

2 Computational Assumptions

In this section, we will first recall some classical computational assumptions and introduce a new one, of independent interest, as it can find many use cases for privacy-preserving protocols.

2.1 Classical Assumptions

All our assumptions will be in the Diffie-Hellman vein, in the pairing setting. We will thus consider an algorithm that, on a security parameter κ, generates $\mathsf{param} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, \mathfrak{g}, e) \leftarrow \mathcal{G}(\kappa)$, an asymmetric pairing setting, with three groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of prime order p (with 2κ bit-length), g is a generator of $\mathbb{G}_1$ and $\mathfrak{g}$ is a generator of $\mathbb{G}_2$. In addition, the application $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerated bilinear map, hence $e(g, \mathfrak{g})$ is also a generator of $\mathbb{G}_T$.

Definition 1 (Discrete Logarithm (DL) Assumption). In a group $\mathbb{G}$ of prime order p, it states that for any generator g, given $y = g^x$, it is computationally hard to recover x.

Definition 2 (Square Discrete Logarithm (SDL) Assumption). In a group $\mathbb{G}$ of prime order p, it states that for any generator g, given $y = g^x$ and $z = g^{x^2}$, it is computationally hard to recover x.

Definition 3 (Twin Discrete Logarithm (TDL) Assumption). In groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order p, it states that for any generators g and $\mathfrak{g}$ of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively, given $f = g^x$ and $\mathfrak{f} = \mathfrak{g}^x$, it is computationally hard to recover x.

Definition 4 (Decisional Diffie-Hellman (DDH) Assumption). In a group $\mathbb{G}$ of prime order p, it states that for any generator g, the two following distributions are computationally indistinguishable:

$$\mathcal{D}_{\mathsf{dh}}(g) = \{(g, g^x, h, h^x); h \xleftarrow{\$} \mathbb{G}, x, \xleftarrow{\$} \mathbb{Z}_p\}$$
$$\mathcal{D}^4_{\$}(g) = \{(g, g^x, h, h^y); h \xleftarrow{\$} \mathbb{G}, x, y, \xleftarrow{\$} \mathbb{Z}_p\}.$$

This is well-know, using an hybrid argument, or the random-self-reducibility, that this assumption implies the Decisional Multi Diffie-Hellman (DMDH) Assumption, which claims the indistinguishability, for any constant $n \in \mathbb{N}$, of the distributions:

$$\mathcal{D}^n_{\mathsf{mdh}}(g) = \{(g, (g^{x_i})_i, h, (h^{x_i})_i); h \xleftarrow{\$} \mathbb{G}, (x_i)_i \xleftarrow{\$} \mathbb{Z}_p^n\}$$
$$\mathcal{D}^{2n+2}_{\$}(g) = \{(g, (g^{x_i})_i, h, (h^{y_i})_i); h \xleftarrow{\$} \mathbb{G}, (x_i)_i, (y_i)_i \xleftarrow{\$} \mathbb{Z}_p^n\}.$$

Definition 5 (Decisional Square Diffie-Hellman (DSDH) Assumption). In a group $\mathbb{G}$ of prime order p, it states that for any generator g, the two following distributions are computationally indistinguishable:

$$\mathcal{D}_{\mathsf{sdh}}(g) = \{(g, g^x, g^{x^2}), x \xleftarrow{\$} \mathbb{Z}_p\} \qquad \mathcal{D}^3_{\$}(g) = \{(g, g^x, g^y), x, y \xleftarrow{\$} \mathbb{Z}_p\}.$$

It is worth noticing that the DSDH Assumption implies the SDL Assumption: if one can break SDL, from g, g^x, g^{x^2}, one can compute x and thus break DSDH.

2.2 Unlinkability Assumption

For anonymity properties, we will use some kind of credential, that can be defined as follows for a scalar u and a basis $g \in \mathbb{G}_1$, with $\mathfrak{g} \in \mathbb{G}_2$, $r, t \in \mathbb{Z}_p$:

$$\mathsf{Cred}(u, g; \mathfrak{g}, r, t) = \left(\begin{pmatrix} g \\ \mathfrak{g} \end{pmatrix}, \begin{pmatrix} g \\ \mathfrak{g} \end{pmatrix}^t, \begin{pmatrix} g \\ g^t \end{pmatrix}^r \times \begin{pmatrix} 1 \\ g^u \end{pmatrix}, \mathfrak{g}^u \right)$$

Definition 6 (Unlinkability Assumption). In groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order p, for any $g \in \mathbb{G}_1$ and $\mathfrak{g} \in \mathbb{G}_2$, with the definition below, it states that the distributions $\mathcal{D}_{g,\mathfrak{g}}(u,u)$ and $\mathcal{D}_{g,\mathfrak{g}}(u,v)$ are computationally indistinguishable, for any $u, v \in \mathbb{Z}_p$:

$$\mathcal{D}_{g,\mathfrak{g}}(u,v) = \left\{ (\mathsf{Cred}(u,g;\mathfrak{g},r,t), \mathsf{Cred}(v,g;\mathfrak{g}',r',t')); \begin{array}{l} \mathfrak{g}' \xleftarrow{\$} \mathbb{G}_2, \\ r,t,r',t' \xleftarrow{\$} \mathbb{Z}_p \end{array} \right\}$$

Intuitively, the third component is an ElGamal ciphertext of the g^u, which hides it, and makes indistinguishable another encryption g^u from an encryption of g^v while, given $(\mathfrak{g}, \mathfrak{g}^u)$ and $(\mathfrak{g}', \mathfrak{g}'^v)$, one cannot guess whether $u = v$, under the DDH assumption in $\mathbb{G}_2$. However the pairing relation allows to check consistency:

$$e(g^{rt} \cdot g^u, \mathfrak{g}) = e(g^r, \mathfrak{g}^t) \cdot e(g, \mathfrak{g}^u) = e(g^r, \mathfrak{g}^t) \cdot e(g, \mathfrak{g})^u$$
$$e(g^{r't'} \cdot g^v, \mathfrak{g}') = e(g^{r'}, \mathfrak{g}'^{t'}) \cdot e(g, \mathfrak{g}'^v) = e(g^{r'}, \mathfrak{g}'^{t'}) \cdot e(g, \mathfrak{g}')^v$$

Because of the independent group elements $\mathfrak{g}$ and $\mathfrak{g}' = \mathfrak{g}^s$ in the two credentials, this assumption clearly holds in the generic bilinear group model, as one would either need to compare $u = v$ or equivalently $rt = r't'$, whereas combinations only lead to $e(g, \mathfrak{g})$ to the relevant powers rt, $sr't'$, as well as u and sv, for an unknown s.

Thanks to this unlinkability assumption, and the randomizability of the above credential, proving knowledge of u can lead to anonymous credentials. But our main application will be for our anonymous shuffle presented in Section 4.

3 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 sub-spaces, 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 groups. Then Libert *et al.* [LPJY13] combined both notions and proposed a linearly-homomorphic signature scheme, that is furthermore structure-preserving. Our work is inspired from this construction.

3.1 Variant of the LPJY Signature

We first adapt the LPJY signature [LPJY13], that was initially designed in the symmetric pairing setting, to the asymmetric setting. The basic construction (the one-time version) can be adapted as follows:

$\mathsf{Keygen}(\kappa)$: Given a security parameter κ, let $\mathsf{param} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, \mathfrak{g}, e) \leftarrow \mathcal{G}(\kappa)$, where g and $\mathfrak{g}$ are random generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. One randomly chooses $(\mathfrak{g}_u, \mathfrak{g}_v, \mathfrak{h}_u, \mathfrak{h}_w) \xleftarrow{\$} \mathbb{G}_2^4$, as well as $\mathsf{sk}_i = (u_i, v_i, w_i) \xleftarrow{\$} \mathbb{Z}_p^3$, for $i = 1, \ldots, n$, which defines the signing key $\mathsf{sk} = (\mathsf{sk}_i)_i$ and the verification key $\mathsf{vk} = (\mathsf{param}, \mathfrak{g}_u, \mathfrak{g}_v, \mathfrak{h}_u, \mathfrak{h}_w, (\mathsf{vk}_i)_i)$ for $\mathsf{vk}_i = (\mathfrak{g}_i = \mathfrak{g}_u^{u_i} \mathfrak{g}_v^{v_i}, \mathfrak{h}_i = \mathfrak{h}_u^{u_i} \mathfrak{h}_w^{w_i})$.

$\mathsf{Sign}(\mathsf{sk}, (M_i)_i)$: Given a signing key $\mathsf{sk} = (u_i, v_i, w_i)_i$ and a vector-message $(M_i)_i \in \mathbb{G}_1^n$, one sets $\sigma = (\sigma_1, \sigma_2, \sigma_3) \in \mathbb{G}_1^3$, where

$$\sigma_1 = \prod_{i=1}^n M_i^{-u_i}, \qquad \sigma_2 = \prod_{i=1}^n M_i^{-v_i}, \qquad \sigma_3 = \prod_{i=1}^n M_i^{-w_i}$$

$\mathsf{Verif}(\mathsf{vk}, (M_i)_i, \sigma)$: Given a verification key $\mathsf{vk} = (\mathsf{param}, \mathfrak{g}_u, \mathfrak{g}_v, \mathfrak{h}_u, \mathfrak{h}_w, (\mathfrak{g}_i, \mathfrak{h}_i)_i)$ and a vector-message $(M_i)_i \in \mathbb{G}_1^n$ with the signature $\sigma = (\sigma_1, \sigma_2, \sigma_3) \in \mathbb{G}_2^3$, one can check whether both equalities hold or not

$$e(\sigma_1, \mathfrak{g}_u) \cdot e(\sigma_2, \mathfrak{g}_v) \cdot \prod_{i=1}^n e(M_i, \mathfrak{g}_i) = 1_{\mathbb{G}_T} = e(\sigma_1, \mathfrak{h}_u) \cdot e(\sigma_3, \mathfrak{h}_w) \cdot \prod_{i=1}^n e(M_i, \mathfrak{h}_i).$$

They proved the unforgeability under the Simultaneous Double Pairing assumption, which is implied by the linear assumption in the symmetric case. But the unforgeability is not the usual notion, because of the linear homomorphism: given several vector-messages with their signatures, this is possible to generate the signature of any linear combination of the vector-messages (actually, in the exponents): given two vector-messages $\boldsymbol{M} = (M_i)_i$ and $\boldsymbol{M}' = (M'_i)_i$, with their signatures $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ and $\sigma' = (\sigma'_1, \sigma'_2, \sigma'_3)$, then $\sigma'' = (\sigma''_1 = {\sigma_1}^{\alpha}{\sigma'_1}^{\beta}, \sigma''_2 = {\sigma_2}^{\alpha}{\sigma'_2}^{\beta}, \sigma''_3 = {\sigma_3}^{\alpha}{\sigma'_3}^{\beta})$ is a valid signature of $\boldsymbol{M}'' = \boldsymbol{M}^{\alpha} \cdot \boldsymbol{M}'^{\beta} = ({M_i}^{\alpha}{M'_i}^{\beta})_i$.

The security proof thus guarantees that no adversary can generate a valid signature for a message outside the linear span (in the exponents) of the already signed messages.

3.2 Improved Construction in the Generic Bilinear Group Model

This linear homomorphism is a very nice property, but this would be good to have the additional information of the explicit linear combination produced from the input vector-messages. This is actually possible when one focuses on algebraic adversaries [EHK+13, FKL18], or in the generic group model [Sho97, BBG05, Boy08]. But in such a model, we can simplify the construction, and prove its security. Then, we will show how such a scheme can be used for privacy-preserving constructions.

$\mathsf{Keygen}(\kappa)$**:** Given a security parameter κ, let $\mathsf{param} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, \mathfrak{g}, e) \leftarrow \mathcal{G}(\kappa)$, where g and $\mathfrak{g}$ are random generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. One randomly chooses $\mathsf{sk}_i = s_i \xleftarrow{\$} \mathbb{Z}_p$, for $i = 1, \ldots, n$, which defines the signing key $\mathsf{sk} = (\mathsf{sk}_i)_i$, and the verification key $\mathsf{vk} = (\mathsf{param}, (\mathfrak{g}_i)_i)$ for $\mathfrak{g}_i = \mathfrak{g}^{s_i}$;

$\mathsf{Sign}(\mathsf{sk}, \boldsymbol{M} = (M_i)_i)$**:** Given a signing key $\mathsf{sk} = (s_i)_i$ and a vector-message $\boldsymbol{M} = (M_i)_i \in \mathbb{G}_1^n$, one sets $\sigma = \prod_{i=1}^n M_i^{s_i} \in \mathbb{G}_1$;

$\mathsf{Verif}(\mathsf{vk}, \boldsymbol{M} = (M_i)_i, \sigma)$**:** Given a verification key $\mathsf{vk} = (\mathsf{param}, (\mathfrak{g}_i)_i)$, a vector-message $\boldsymbol{M} = (M_i)_i$, and a signature σ, one checks whether the equality $e(\sigma, \mathfrak{g}) = \prod_{i=1}^n e(M_i, \mathfrak{g}_i)$ holds or not.

As above, this signature scheme is linearly-homomorphic, but it provides a few interesting properties, that will be exploited later:

Property 7 (Message Homomorphism). Given several vector-messages with their signatures, this is possible to generate the signature of any linear combination of the vector-messages: given two vectors $\boldsymbol{M} = (M_i)_i$ and $\boldsymbol{M} = (M'_i)_i$, with their signatures σ and σ', then $\sigma'' = \sigma^{\alpha}\sigma'^{\beta}$ is a valid signature of $\boldsymbol{M}'' = \boldsymbol{M}^{\alpha}\boldsymbol{M}'^{\beta}$, where the operations are component-wise.

In addition, some relations hold on the keys:

Property 8 (Key Homomorphism). It is important to precise the basis $\mathfrak{g}$ used in the verification process: if $e(\sigma, \mathfrak{g}) = \prod_{i=1}^n e(M_i, \mathfrak{g}_i)$ holds, then $e(\sigma, \mathfrak{g}^{\alpha}) = \prod_{i=1}^n e(M_i, \mathfrak{g}_i^{\alpha})$ holds too. Then, if a message-signature is valid for a verification key vk, with respect to a basis $\mathfrak{g}$, then it is also valid for the verification key $\mathsf{vk}' = \mathsf{vk}^{\alpha}$, with respect to the basis $\mathfrak{g}' = \mathfrak{g}^{\alpha}$, for any α.

Furthermore as $\prod_{i=1}^n e(M_i, \mathfrak{g}_i^{\alpha} \cdot \mathfrak{g}_i'^{\beta}) = \prod_{i=1}^n e(M_i, \mathfrak{g}_i)^{\alpha} \cdot e(M_i, \mathfrak{g}'_i)^{\beta}$, $\mathsf{Sign}(\alpha \cdot \mathsf{sk} + \beta \cdot \mathsf{sk}', \boldsymbol{M}) = \mathsf{Sign}(\mathsf{sk}, \boldsymbol{M})^{\alpha} \cdot \mathsf{Sign}(\mathsf{sk}', \boldsymbol{M})^{\beta}$.

3.3 Unforgeability

First, we consider adversaries in the generic bilinear group model [BBG05], and show that for any new valid pair $(\boldsymbol{M}, \sigma)$, one necessarily knows how $\boldsymbol{M}$ is linearly related to the already known signed vector-messages.

Theorem 9. *In the generic bilinear group model, given n valid pairs $(\boldsymbol{M}_j = (M_{j,i})_i, \sigma_j)_j$ under a verification key vk, for any adversary that produces a new valid pair $(\boldsymbol{M} = (M_i)_i, \sigma)$ under the same verification key vk, there exits $(\alpha_j)_j$ such that $\boldsymbol{M} = \prod_j \boldsymbol{M}_j^{\alpha_j}$ and $\sigma = \prod_j \sigma_j^{\alpha_j}$.*

Proof. The adversary is given $(\boldsymbol{M}_j = (M_{j,i})_i, \sigma_j)_j$ which contains group elements in $\mathbb{G}_1$, as well as the verification key $\mathsf{vk} = (\mathfrak{g}_k)_k$ in $\mathbb{G}_2$. For any combination query, the simulator will consider the input elements as independent variables $X_{j,i}$, V_j, and $\mathfrak{S}_k$ to formally represent the discrete logarithms of $M_{j,i}$ and σ_i in basis g, and $\mathfrak{g}_k$ in basis $\mathfrak{g}$. As usual, any new element can be seen as a multivariate polynomial in these variables, of degree maximal 2 (when there is a mix between $\mathbb{G}_1$ and $\mathbb{G}_2$ group elements). If two elements correspond to the same polynomial, they are definitely equal, and the simulator will provide the same representation. If two elements correspond to different polynomials, the simulator will provide random independent representations. The view of the adversary remains unchanged unless the actual instantiations would make the representations equal: they would be equal with probability at most $2/p$, when the variables are set to random values. After N combination queries, we have at most $N^2/2$ pairs of different polynomials that might lead to a collision for a random setting with probability less than N^2/p. Excluding such collisions, we can thus consider the polynomial representations only, denoted $\sim$. Then, for the output $(\boldsymbol{M} = (M_k)_k, \sigma)$, one knows $\alpha_{k,j,i}, \beta_{k,j}, \gamma_{i,j}, \delta_j$, such that:

$$M_k \sim \sum_{j,i} \alpha_{k,j,i} X_{j,i} + \sum_j \beta_{k,j} V_j \qquad\qquad \sigma \sim \sum_{j,i} \gamma_{j,i} X_{j,i} + \sum_j \delta_j V_j.$$

As $((M_{j,i})_i, \sigma_j)_j$ and $((M_k)_k, \sigma)$, are valid input and output pairs, we have the following relations between polynomials:

$$V_j = \sum_i X_{j,i} \mathfrak{S}_i \qquad \sum_{j,i} \gamma_{j,i} X_{j,i} + \sum_j \delta_j V_j = \sum_k \left(\sum_{j,i} \alpha_{k,j,i} X_{j,i} + \sum_j \beta_{k,j} V_j \right) \mathfrak{S}_k$$
$$= \sum_{k,j,i} \alpha_{k,j,i} X_{j,i} \mathfrak{S}_k + \sum_{k,j} \beta_{k,j} V_j \mathfrak{S}_k$$

Hence, the two polynomials are equal:

$$\sum_{j,i} \gamma_{j,i} X_{j,i} + \sum_{j,i} (\delta_j - \alpha_{i,j,i}) X_{j,i} \mathfrak{S}_i = \sum_{k \neq i,j,i} \alpha_{k,j,i} X_{j,i} \mathfrak{S}_k + \sum_{k,j} \beta_{k,j} V_j \mathfrak{S}_k$$

which leads, for all i, j, to $\gamma_{j,i} = 0$ and $\delta_j = \alpha_{i,j,i}$, and for $k \neq i$, $\alpha_{k,j,i} = 0$ and $\beta_{k,j} = 0$. Hence, $M_k \sim \sum_j \delta_j X_{j,k}$ and $\sigma \sim \sum_j \delta_j V_j$, which means that we have $(\delta_j)_j$ such that $M_k = \prod_j M_{j,k}^{\delta_j}$ and $\sigma = \prod_j \sigma_j^{\delta_j}$. □

3.4 Notations and Constraints

As one can derive signatures of linear combinations of the inputs, from signed inputs, one can talk about signing sub-vector spaces. Note that linear computations are seen in the exponents. Since we will mainly work on sub-vector spaces of dimension 2, we will denote $\sigma = \mathsf{Sign}(\mathsf{sk}, (\boldsymbol{M}, \boldsymbol{M'}))$, with the verification check $\mathsf{Verif}(\mathsf{vk}, \sigma, (\boldsymbol{M}, \boldsymbol{M'})) = 1$, a signature that allows to derive a valid σ' for any linear combinations of $\boldsymbol{M}$ and $\boldsymbol{M'}$. In general, this can be the concatenation of $\sigma_1 = \mathsf{Sign}(\mathsf{sk}, \boldsymbol{M})$ and $\sigma_2 = \mathsf{Sign}(\mathsf{sk}, \boldsymbol{M'})$, but some joint random coins may be needed, and some common elements can be merged, for efficiency reasons, as it will be shown in the second instantiation below.

We will also be interested in signing affine spaces: given a signature on $\boldsymbol{M}$ and $\boldsymbol{N}$, one wants to limit signatures on $\boldsymbol{M} \times \boldsymbol{N}^\alpha$ and $1 \times \boldsymbol{N}^\beta$. This is possible by expanding the messages with one more component: for $\overline{\boldsymbol{M}} = (g, \boldsymbol{M})$ and $\overline{\boldsymbol{N}} = (1, \boldsymbol{N})$, linear combinations are of the form $(g^\alpha, \boldsymbol{M}^\alpha \boldsymbol{N}^\beta)$. By imposing the first component to be g, one limits to $\alpha = 1$, and thus to $(g, \boldsymbol{M}\boldsymbol{N}^\beta) = \overline{\boldsymbol{M}} \times \overline{\boldsymbol{N}}^\beta$, while by imposing the first component to be 1, one limits to $\alpha = 0$, and thus to $(1, \boldsymbol{N}^\beta) = \overline{\boldsymbol{N}}^\beta$.

3.5 Non-Miscibility

Once we know only linear combinations are possible among signed vector-messages, one may require stronger restrictions such as avoiding combinations between families of vectors. In our case, while ciphertexts will be signed under different voter's keys, all the voter's keys will be signed under the same authority's key. Then, it will not be possible to combine the ciphertexts of the different voters (signed under different keys). However, the voter's keys might be combined, which could lead to a new illegitimate voter.

We thus need some notion of *non-miscibility*, which prevents combinations between some vectors. In [LPJY13], they named *one-time* linearly-homomorphic the previous scheme, where all the signed-vectors can be combined, and *full-fledged* linearly-homomorphic a scheme, where one specifies with labels the families of vectors that can be combined together. Again, since we target efficiency, with security analysis in the generic bilinear group model, we can use more efficient approaches.

Non-Miscibility of Vectors. First, when one wants to avoid any miscibility, and just allow to convert a signature of $\boldsymbol{M}$ into a signature of $\boldsymbol{M}^\alpha$, while they are all of the same format, one can use expanded vectors (as in Section 3.4), by concatenating a vector that satisfies this restriction: from multiple distinct Square Diffie-Hellman tuples $(g_i, g_i^{w_i}, g_i^{w_i^2})$, a linear combination that is also a Square Diffie-Hellman tuple cannot use more than one input tuple. We prove it in two different cases: with random and independent bases g_i, but possibly public w_i's, or with a common basis $g_i = g$, but secret w_i's. More precisely, we can state the following theorems, which proofs can be found in the Appendix A.

We stress that in the first theorem, the w_i's are random and public (assumed distinct), but the bases g_i's are truly randomly and independently generated.

Theorem 10. *Given n valid Square Diffie-Hellman tuples $(g_i, a_i = g_i^{w_i}, b_i = a_i^{w_i})$, with w_i, for random $g_i \xleftarrow{\$} \mathbb{G}$ and $w_i \xleftarrow{\$} \mathbb{Z}_p$, outputting $(\alpha_i)_{i=1,\ldots,n}$ such that $(G = \prod g_i^{\alpha_i}, A = \prod a_i^{\alpha_i}, B = \prod b_i^{\alpha_i})$ is a valid Square Diffie-Hellman, with at least two non-zero coefficients α_i, is computationally hard under the* DL *assumption.*

In the second scenario, the basis is common (for all i, $g_i = g$), but the w_i's are secret, still random and thus assumed distinct.

Theorem 11. *Given n valid Square Diffie-Hellman tuples $(g, a_i = g^{w_i}, b_i = a_i^{w_i})$ for any $g \in \mathbb{G}$ and random $w_i \xleftarrow{\$} \mathbb{Z}_p$, outputting $(\alpha_i)_{i=1,\ldots,n}$ such that $(G = \prod g^{\alpha_i}, A = \prod a_i^{\alpha_i}, B = \prod b_i^{\alpha_i})$ is a valid Square Diffie-Hellman, with at least two non-zero coefficients α_i, is computationally hard under the* SDL *assumption.*

Efficient Non-Miscible Linearly-Homomorphic Signature. Another approach is the use of the scheme proposed in [FHS19]. We can describe it as follows, in the similar vein as our previous construction:

$\mathsf{Keygen}^*(\kappa)$**:** Given a security parameter κ, let $\mathsf{param} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, \mathfrak{g}, e) \leftarrow \mathcal{G}(\kappa)$, where g and $\mathfrak{g}$ are random generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. One randomly chooses $\mathsf{sk}_i = s_i \xleftarrow{\$} \mathbb{Z}_p$, for $i = 1, \ldots, n$, which defines the signing key $\mathsf{sk} = (\mathsf{sk}_i)_i$, and the verification key $\mathsf{vk} = (\mathsf{param}, (\mathfrak{g}_i)_i)$ for $\mathfrak{g}_i = \mathfrak{g}^{s_i}$;

$\mathsf{Sign}^*(\mathsf{sk}, \boldsymbol{M} = (M_i)_i, \lambda)$**:** Given a signing key $\mathsf{sk} = (s_i)_i$ and a vector-message $\boldsymbol{M} = (M_i)_i \in \mathbb{G}_1^n$, together with some label λ to which the signer associates a private random scalar $R_\lambda \xleftarrow{\$} \mathbb{Z}_p$, one sets $(\sigma = (\prod_{i=1}^n M_i^{s_i})^{R_\lambda}, \tau_1 = g^{1/R_\lambda}, \tau_2 = \mathfrak{g}^{1/R_\lambda})$;

$\mathsf{Verif}^*(\mathsf{vk}, (\sigma, \tau_1, \tau_2), \boldsymbol{M} = (M_i)_i)$**:** Given a verification key $\mathsf{vk} = (\mathsf{param}, (\mathfrak{g}_i)_i)$, a vector-message $\boldsymbol{M} = (M_i)_i$, and a signature (σ, τ_1, τ_2), one checks if the equalities $e(\sigma, \tau_2) = \prod_{i=1}^n e(M_i, \mathfrak{g}_i)$ and $e(\tau_1, \mathfrak{g}) = e(g, \tau_2)$ hold or not.

When the random values are all privately and randomly chosen, independently for each signature, unforgeability has been proven in [FHS19]. The intuition is the following: first, under the Knowledge of Exponent Assumption [Dam92, HT98, Gro10], that holds in the generic group model, from a new pair (τ_1, τ_2) (on the input of either $(g, \mathfrak{g})$ or another pair), one can extract the common exponent in the two components, from the input pair to the output pair, but no mixing is possible. Then, one can see σ as the signature with the secret key $(R_\lambda s_i)_i$, with respect to $\tau_2 = \mathfrak{g}^{1/R_\lambda}$, instead of $\mathfrak{g}$ in the previous construction.

However, if one knows two signatures (σ, τ_1, τ_2) and $(\sigma', \tau_1, \tau_2)$ on $\boldsymbol{M}$ and $\boldsymbol{M'}$ respectively, both with the same R_λ (and thus the same (τ_1, τ_2)), then $(\sigma^\alpha \sigma'^\beta, \tau_1^{\alpha+\beta}, \tau_2^{\alpha+\beta})$ is a valid signature of $\boldsymbol{M}^\alpha \boldsymbol{M'}^\beta$, still with the same R_λ: this is thus a linearly-homomorphic signature scheme, where one can control the families of messages that can be combined.

In addition, one can easily randomize R_λ: from a signature (σ, τ_1, τ_2) on $\boldsymbol{M}$ for R_λ, $(\sigma^R, \tau_1^{1/R}, \tau_2^{1/R})$ is a new signature on $\boldsymbol{M}$ for $R \cdot R_\lambda$, and so a totally unrelated random value. This signature does not allow any further combination.

As already explained above, we will essentially work on sub-vector spaces of dimension 2: we will thus denote $\sigma = (\sigma_1, \sigma_2, \tau_1, \tau_2) = \mathsf{Sign}^*(\mathsf{sk}, (\boldsymbol{M}, \boldsymbol{M'})) \in \mathbb{G}_1^3 \times \mathbb{G}_2$, where $(\sigma_1, \tau_1, \tau_2) = \mathsf{Sign}^*(\mathsf{sk}, \boldsymbol{M}, \lambda)$ and $(\sigma_2, \tau_1, \tau_2) = \mathsf{Sign}^*(\mathsf{sk}, \boldsymbol{M'}, \lambda)$, for a common random label λ, and thus a common random R_λ, which leads to the same τ_1 and τ_2.

Note that in the following, the use of this signature scheme will switch $\mathbb{G}_1$ and $\mathbb{G}_2$, as the messages to be signed will be the verification keys of the previous signature scheme, and thus in $\mathbb{G}_2$. Then the verification keys of this scheme will be in $\mathbb{G}_1$. We focus on this instantiation in the body of the paper, while the construction with the Square Diffie-Hellman tuples is studied in the Appendices.

4 Mix-Networks

A major tool for anonymity is mix-nets, a network of mix-servers [Cha81], that allows to shuffle ciphertexts so that all the input ciphertexts are in the output, but cannot be linked together. Then, one can safely decrypt the output ciphertexts without leaking any information about the sender. This has many applications such as anonymous routing and electronic voting. Whereas it is easy for a server to apply a random permutation on ciphertexts and randomize them, it is not that easy to provide a proof of correctness that is publicly verifiable, and compact. With our Linearly-Homomorphic Signatures, it is possible to provide such a proof in an efficient and scalable way.

4.1 Introduction

In the context of electronic voting, each vote is encrypted under the system public key EK into C_i and authenticated by its sender under its public key vk_i into a ballot $\mathcal{B}_i$. The ballot-box collects all of them: $\mathcal{BB}\mathsf{ox} = \{\mathcal{B}_i\}$. The mix-server can randomize the ballots (and namely the ciphertexts C_i into C'_i) and permute them into the new ballot-box $\mathcal{BB}\mathsf{ox}' = \{\mathcal{B}'_i\}$. The goal of the proof is to show the existence of a permutation Π such that for every i, ballot $\mathcal{B}'_i$ is a randomization of ballot $\mathcal{B}_{\Pi(i)}$. Then, the output ballot-box can be mixed again by another mix-server, hence the notion of mix-network. While the soundness of the proof of mixing should guarantee the existence of the permutation Π, the zero-knowledge property is also required to guarantee that no information leaks about Π.

4.2 General Description

The high-level description is presented in Figure 1, with

Parameters

For a security parameter κ, let $\mathsf{Gparam} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, \mathfrak{g}, e) \leftarrow \mathcal{G}(\kappa)$ be the global parameters, where g and $\mathfrak{g}$ are random generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively;

Keys

- System keys: $(\mathsf{SK}, \mathsf{VK}) \leftarrow \mathsf{Keygen}^*(\kappa)$ is a pair of signing-verification keys of messages in $\mathbb{G}_2^5$ (a linearly-homomorphic signature scheme, with non-miscibility guarantees) and $\mathsf{DK} = d \xleftarrow{\$} \mathbb{Z}_p$, $\mathsf{EK} = h = g^d$ the ElGamal encryption key;
- Round key: $\mathsf{vk}_r = (1, \mathfrak{g}_r, 1, 1) \in \mathbb{G}_2^4$ and $\overline{\mathsf{vk}}_r = (1, \mathsf{vk}_r) \in \mathbb{G}_2^5$ the expanded key, to be signed under SK, with respect to g;
- Sender keys: $\mathsf{sk}_i = (u_i, v_i, x_i, y_i) \in \mathbb{Z}_p^4$ the signing key, $\mathsf{vk}_i = (\mathfrak{f}_i = \mathfrak{g}_r^{u_i}, \mathfrak{l}_i = \mathfrak{g}_r^{v_i}, \mathfrak{g}_i = \mathfrak{g}_r^{x_i}, \mathfrak{h}_i = \mathfrak{g}_r^{y_i}) \in \mathbb{G}_2^4$ the associated verification key, to sign messages in $\mathbb{G}_1^4$ (our linearly-homomorphic signature scheme) and $\overline{\mathsf{vk}}_i = (\mathfrak{g}_r, \mathsf{vk}_i) \in \mathbb{G}_2^5$ the expanded verification key to be signed under SK, and the signature of the pairs $(\overline{\mathsf{vk}}_i, \overline{\mathsf{vk}}_r)$: $\Sigma_i = \mathsf{Sign}^*(\mathsf{SK}, (\overline{\mathsf{vk}}_i, \overline{\mathsf{vk}}_r))$ so that, with respect to g,

$$\mathsf{Verif}^*(\mathsf{VK}, (\overline{\mathsf{vk}}_i, \overline{\mathsf{vk}}_r), \Sigma_i) = 1 \tag{1}$$

Ciphertexts

- Global ciphertext: $C_0 = \mathsf{Encrypt}(\mathsf{EK}, 1) = (g, h) \in \mathbb{G}_1^2$ and, for $\ell \xleftarrow{\$} \mathbb{G}_1$, the expanded ciphertext $\overline{C}_0 = (1, \ell, C_0) \in \mathbb{G}_1^4$;
- For each ballot: $C_i = \mathsf{Encrypt}(\mathsf{EK}, M_i) \in \mathbb{G}_1^2$ and, for $\ell_i \xleftarrow{\$} \mathbb{G}_1$, the expanded ciphertext $\overline{C}_i = (g, \ell_i, C_i) \in \mathbb{G}_1^4$, and then the signature of pairs $(\overline{C}_i, \overline{C}_0)$ under sk_i: $\sigma_i = \mathsf{Sign}(\mathsf{sk}_i, (\overline{C}_i, \overline{C}_0))$ so that, with respect to $\mathfrak{g}_r$,

$$\mathsf{Verif}(\mathsf{vk}_i, (\overline{C}_i, \overline{C}_0), \sigma_i) = 1 \tag{2}$$

Mixing

- New round key and ciphertext with $\alpha \xleftarrow{\$} \mathbb{Z}_p$: $\mathfrak{g}'_r = \mathfrak{g}_r^\alpha$

$$\mathsf{vk}'_r = \mathsf{vk}_r^\alpha \qquad\qquad \overline{\mathsf{vk}}'_r = \overline{\mathsf{vk}}_r^\alpha = (1, 1, \mathfrak{g}'_r, 1, 1)$$

- For each ballot, new verification key and ciphertext with $\gamma_i, \delta_i \xleftarrow{\$} \mathbb{Z}_p$:

$$\mathsf{vk}'_i = (\mathsf{vk}_i \cdot \mathsf{vk}_r{}^{\delta_i})^\alpha \qquad \overline{\mathsf{vk}}'_i = (\overline{\mathsf{vk}}_i \cdot \overline{\mathsf{vk}}_r{}^{\delta_i})^\alpha \qquad C'_i = C_i \cdot C_0^{\gamma_i} \qquad \overline{C}'_i = \overline{C}_i \cdot \overline{C}_0^{\gamma_i}$$

- New signatures Σ'_i on $(\overline{\mathsf{vk}}'_i, \overline{\mathsf{vk}}'_r)$ under VK with respect to g, and σ'_i on $(\overline{C}'_i, \overline{C}_0)$ under vk'_i with respect to $\mathfrak{g}'_r$, thanks to the homomorphic properties of the signature schemes (and possibly the randomizability) and (Σ_i, σ_i);
- Zero-knowledge proof π' of Diffie-Hellman tuple for $(\mathfrak{g}_r, \mathfrak{g}'_r, \prod \mathfrak{f}_i, \prod \mathfrak{f}'_i)$;

Then all the new tuples are outputted in random order together with the proof Π.

Verification

The verifier will check the proof π', the number of input tuples is the same as the number of output tuples, the $\mathfrak{f}'_i$ are all distinct, and the equations (1) and (2), are satisfied on individual output tuples.

Figure 1. Shuffling of ElGamal Ciphertexts

– any linearly-homomorphic signature scheme $(\mathsf{Keygen}^*, \mathsf{Sign}^*, \mathsf{Verif}^*)$ that additionally guarantees non-miscibility, that will be used to sign pairs of *expanded* keys in $\mathbb{G}_2$;
– and our linearly-homomorphic signature scheme $(\mathsf{Keygen}, \mathsf{Sign}, \mathsf{Verif})$, that will be used to sign pairs of *expanded* ElGamal ciphertexts in $\mathbb{G}_1$.

For the latter scheme, we will exploit both the message homomorphism (Property 7) and the key homomorphism (Property 8). But let us first give the intuition without expanded vectors, then expansions will be justified by the constraints we will have to impose.

Each sender gets a pair $(\mathsf{sk}_i, \mathsf{vk}_i) \leftarrow \mathsf{Keygen}(\kappa)$ to sign vectors in $\mathbb{G}_1^4$. To vote, the sender first encrypts his vote M_i under an ElGamal encryption scheme, with encryption key EK and signs it to obtain the signed-encrypted ballot (C_i, σ_i) under vk_i with respect to $\mathfrak{g}_r$. But some guarantees are needed.

In order to be sure that a ballot is legitimate, all the verification keys must be certified by the system that signs vk_i under SK, where $(\mathsf{SK}, \mathsf{VK}) \leftarrow \mathsf{Keygen}^*(\kappa)$, into Σ_i. Then, anyone can verify the certified keys $(\mathsf{vk}_i, \Sigma_i)_i$ are valid under the system verification key VK.

For an input ballot box $(\mathsf{vk}_i, \Sigma_i, C_i, \sigma_i)_i$, we want to output a new ballot-box $(\mathsf{vk}'_i, \Sigma'_i, C'_i, \sigma'_i)_i$ so that there exists a permutation Π such that, for each input ballot $(\mathsf{vk}_i, \Sigma_i, C_i, \sigma_i)$, there is an output ballot $(\mathsf{vk}'_j, \Sigma'_j, C'_j, \sigma'_j)$ with $j = \Pi(i)$, where C'_j a randomization of C_i, the signature σ'_j is a valid signature on C'_j under vk'_j, with respect to $\mathfrak{g}'_r$, and vk'_j is appropriately signed in Σ'_j.

The soundness of the proof will crucially rely on the unforgeability properties of the signature schemes. About the privacy, since it is encrypted, the vote is protected, but this is not enough as it will be decrypted in the end. One also needs to guarantee unlinkability between the input and output ballots. As they contain the ciphertexts C_i and C'_j, as well as the verification keys vk_i and vk'_j, they must be transformed in an unlinkable way. To fix that, C'_j must be a randomization of C_i, but also vk'_j must be a randomization of vk_i. Then, the signatures must be adapted, in an unrelated way.

4.3 Randomization

In order to randomize the keys and the ciphertexts, we will exploit the homomorphic signatures on vector spaces, but on expanded versions (denoted with a bar above the elements, as already done previously). The additional components in the expanded keys and ciphertexts will allow to impose constraints on the linear combinations. We then sign the expanded vectors.

Ciphertexts. In order to be able to randomize the ciphertext C_i, the system provides a trivial encryption of 1 (in C_0). With that and thanks to the homomorphic property of the encryption, $C'_i = C_i \cdot C_0^{\gamma_i}$ is a randomization of C_i, for a random user-based scalar γ_i. If $\sigma_i = \mathsf{Sign}(\mathsf{sk}_i, (C_i, C_0))$ with respect to $\mathfrak{g}_r$, σ_i allows to generate a signature of (C'_i, C_0) under sk_i, still with respect to $\mathfrak{g}_r$.

But it would be possible to sign $C'_i = C_i^{\nu_i} \cdot C_0^{\gamma_i}$. Hence, we expand and sign $\overline{C}_i = (g, \ell_i, C_i)$ and $\overline{C}_0 = (1, \ell, C_0)$: the first component of $\overline{C}'_i$ must still be g (see below for the second components). This enforces the correct generation of the randomizations under the unforgeability of our signature scheme, as already explained in Section 3.4: the signature of $(\overline{C}_i, \overline{C}_0)$ corresponds to a signature $\sigma_{i,1}$ of $\overline{C}_i$ and a second signature $\sigma_{i,2}$ of $\overline{C}_0$ both under the secret key sk_i, with respect to $\mathfrak{g}_r$. In other words, $\mathsf{Verif}(\mathsf{vk}_i, \overline{C}_i, \sigma_{i,1}) = 1$ and $\mathsf{Verif}(\mathsf{vk}_i, \overline{C}_0, \sigma_{i,2}) = 1$, with respect to $\mathfrak{g}_r$. Then, from Property 7, $\mathsf{Verif}(\mathsf{vk}_i, \overline{C}_i \cdot \overline{C}_0^{\gamma_i}, \sigma_{i,1} \cdot \sigma_{i,2}^{\gamma_i}) = 1$ with respect to $\mathfrak{g}_r$, hence $\sigma'_{i,1} = \sigma_{i,1} \cdot \sigma_{i,2}^{\gamma_i}$ is a valid signature under sk_i with respect to $\mathfrak{g}_r$. Since we sign vectors $\overline{C}_i$ and $\overline{C}_0$ in $\mathbb{G}_1^4$, $\mathsf{vk}_i = (\mathfrak{f}_i, \mathfrak{l}_i, \mathfrak{g}_i, \mathfrak{h}_i) \in \mathbb{G}_2^4$.

Keys. However, we already explained we also need to randomize vk_i. The first intuition would be to replace $\mathsf{vk}'_i = \mathsf{vk}_i^{\alpha}$, with a global random power α. Under the DDH assumption, the output

keys are unlinkable to the input keys. However, with $\mathfrak{g}'_r = \mathfrak{g}_r^\alpha$: $\mathsf{Verif}(\mathsf{vk}'_i, \overline{C}_i, \sigma_{i,1}) = 1$ still holds with respect to $\mathfrak{g}'_r$, from Property 8. Thus, we add a round key $\mathsf{vk}_r = (1, \mathfrak{g}_r, 1, 1) \in \mathbb{G}_2$ to be able to randomize vk_i by multiplying it by ${\mathsf{vk}_r}^{\delta_i}$. Thereby, vk'_i becomes $(\mathsf{vk}_i \cdot {\mathsf{vk}_r}^{\delta_i})^\alpha$, for a global random scalar α, but a random user-based scalar δ_i, and we can still get the signature of vk'_i under SK. To limit to this transformation for vk'_i, under the unforgeability of the signature $(\mathsf{Keygen}^*, \mathsf{Sign}^*, \mathsf{Verif}^*)$, we also artificially create $\overline{\mathsf{vk}}_r = (1, \mathsf{vk}_r)$ and $\overline{\mathsf{vk}}_i = (\mathfrak{g}_r, \mathsf{vk}_i)$, as the latter must become $(\mathfrak{g}_r^\alpha, \mathsf{vk}_i^\alpha \mathsf{vk}_r^{\mu_i})$, for some β_i, by enforcing the first component to be the new round parameter $\mathfrak{g}'_r = \mathfrak{g}_r^\alpha$: $\mu_i = \alpha\delta_i$.

Impact on Signatures. Then, from Property 8, as both $\mathsf{Verif}(\mathsf{vk}_i, \overline{C}'_i, \sigma_{i,1} \cdot \sigma_{i,2}^{\gamma_i}) = 1$ and $\mathsf{Verif}(\mathsf{vk}_i, \overline{C}_0, \sigma_{i,2}) = 1$, with respect to $\mathfrak{g}_r$, we also have $\mathsf{Verif}(\mathsf{vk}_i \cdot \mathsf{vk}_r^{\delta_i}, \overline{C}'_i, \sigma_{i,1} \cdot \sigma_{i,2}^{\gamma_i} \cdot {\ell'_i}^{\delta_i}) = 1$ and $\mathsf{Verif}(\mathsf{vk}_i \cdot \mathsf{vk}_r^{\delta_i}, \overline{C}_0, \sigma_{i,2} \cdot {\ell'_i}^{\delta_i}) = 1$, with respect to $\mathfrak{g}_r$. Eventually, $\mathsf{Verif}(\mathsf{vk}'_i, \overline{C}'_i, \sigma_{i,1} \cdot \sigma_{i,2}^{\gamma_i} \cdot {\ell'_i}^{\delta_i}) = 1$ and $\mathsf{Verif}(\mathsf{vk}'_i, \overline{C}_0, \sigma_{i,2} \cdot {\ell'_i}^{\delta_i}) = 1$ with respect to $\mathfrak{g}'_r$, hence $\sigma'_{i,1} = \sigma_{i,1} \cdot \sigma_{i,2}^{\gamma_i} \cdot {\ell'_i}^{\delta_i}$ and $\sigma'_{i,2} = \sigma_{i,2} \cdot {\ell'_i}^{\delta_i}$

The role of $\mathsf{vk}_r = (1, \mathfrak{g}_r, 1, 1)$ is to prevent the above signature invariant by breaking the linearity and introducing randomness in the second element of the key. Here comes the explanation of ℓ_i and ℓ in respectively $\overline{C}_i$ and $\overline{C}_0$. The goal of the next section will be to formally prove this intuition.

4.4 Diffie-Hellman Proof

From the above randomizations, we will be able to prove unlinkability (a.k.a. zero-knowledge) property, and from the unforgeability properties of the signature schemes, we know that all the vk'_i correspond to legitimate signers, as they need to be valid under VK into Σ'_i. Each new key comes

- either from a vk_i in the initial ballot-box, and unless the voter colludes, it signs a randomization of the initial ciphertext;
- or from a new legitimate vk_i, that was not in the initial ballot-box. The corresponding legitimate voter must collude to provide a new ballot.

Let us denote $\mathfrak{P} = \prod \mathfrak{f}_i = \mathfrak{g}_r^{\sum u_i}$, on all the input ballots. Let us denote $\mathfrak{P}' = \prod \mathfrak{f}'_i = \mathfrak{g}'_r{}^{\sum u_i}$, on all the output ballots. If the input and output ballot-boxes contain the same ballots (with the same secret u_i), then $\mathfrak{P}' = \mathfrak{P}^\alpha$. Otherwise, if one checks the numbers of ballots are the same, an honest ballot from the input ballot-box must be removed: one can break the discrete logarithm (as we will formally prove) if one additionally checks $\mathfrak{P}' = \mathfrak{P}^\alpha$. Hence, by adding a proof of Diffie-Hellman tuple for $(\mathfrak{g}_r, \mathfrak{g}'_r, \mathfrak{P}, \mathfrak{P}')$, together with the same number of ballots, one has the guarantee the same voters are represented in the two ballot-boxes, and thus their ciphertexts have been randomized.

We could use a Schnorr-like non-interactive zero-knowledge proof of Diffie-Hellman tuple, but we will instead use a Groth-Sahai proof, as such proofs can be combined together into a unique one after multiple mixing steps: let $v_{1,1}, v_{1,2}, v_{2,1}, v_{2,2} \in \mathbb{G}_1$, such that $(v_{1,1}, v_{1,2}, v_{2,1}, g \times v_{2,2})$ is not a Diffie-Hellman tuple. With a commitment of α: $\mathsf{Com} = (c = v_{2,1}^\alpha v_{1,1}^\mu, d = v_{2,2}^\alpha v_{1,2}^\mu g^\alpha)$, for a random $\mu \xleftarrow{\$} \mathbb{Z}_p$, one can set $\Theta = \mathfrak{g}_r^\mu$ and $\Psi = \mathfrak{P}^\mu$, which satisfy

$$e(c, \mathfrak{g}_r) = e(v_{2,1}, \mathfrak{g}'_r) \cdot e(v_{1,1}, \Theta) \qquad e(d, \mathfrak{g}_r) = e(v_{2,2} \cdot g, \mathfrak{g}'_r) \cdot e(v_{1,2}, \Theta)$$
$$e(c, \mathfrak{P}) = e(v_{2,1}, \mathfrak{P}') \cdot e(v_{1,1}, \Psi) \qquad e(d, \mathfrak{P}) = e(v_{2,2} \cdot g, \mathfrak{P}') \cdot e(v_{1,2}, \Psi)$$

This proof is zero-knowledge, under the DDH assumption in $\mathbb{G}_1$: by switching $(v_{1,1}, v_{1,2}, v_{2,1}, g \times v_{2,2})$ into a Diffie-Hellman tuple, one can simulate the proof.

In addition, as explained on Figure 2, in a new mixing step, one can just update the proof: this way, we obtain a constant-size overhead, whatever the number of rounds.

From round k to round $k+1$ (with the use of Groth-Sahai proofs)

1. one computes $\mathcal{BB}\mathsf{ox}^{(k+1)}$ from $\mathcal{BB}\mathsf{ox}^{(k)}$ with the Mixing of a shuffle with α;
2. from the Diffie-Hellman proof $\pi^{(k)}$ for $(\mathfrak{g}_r^{(0)}, \mathfrak{g}_r^{(k)}, \mathfrak{P}^{(0)}, \mathfrak{P}^{(k)} = \prod \mathfrak{f}_i^{(k)})$,

$$\mathsf{Com}^{(k)} = (c = v_{2,1}^{\alpha_k} v_{1,1}^{\mu_k}, d = v_{2,2}^{\alpha_k} v_{1,2}^{\mu_k} g^{\alpha_k}) \qquad \Theta^{(k)} = (\mathfrak{g}_r^{(k)})^{\mu_k} \qquad \Psi^{(k)} = (\mathfrak{P}^{(k)})^{\mu_k}$$

one computes the proof $\pi^{(k+1)}$ for $(\mathfrak{g}_r^{(0)}, \mathfrak{g}_r^{(k+1)}, \mathfrak{P}^{(0)}, \mathfrak{P}^{(k+1)})$, for $\mu \xleftarrow{\$} \mathbb{Z}_p$:

$$\mathsf{Com}^{(k+1)} = (c^{\alpha} \cdot v_{1,1}^{\mu}, d^{\alpha} \cdot v_{1,2}^{\mu})$$
$$\Theta^{(k+1)} = (\Theta^{(k)})^{\alpha} \cdot (\mathfrak{g}_r^{(k)})^{\mu} \qquad \Psi^{(k+1)} = (\Psi^{(k)})^{\alpha} \cdot (\mathfrak{P}^{(k+1)})^{\mu}$$

3. one destroys $\mathcal{BB}\mathsf{ox}^{(k)}$.

Last round N

For a mix-net of N rounds, one just have to keep: $\mathcal{BB}\mathsf{ox}^{(0)}, \mathcal{BB}\mathsf{ox}^{(N)}$ and the global proof $(\mathsf{Com}^{(N)}, \Theta^{(N)}, \Psi^{(N)})$.

Figure 2. Efficient Mix-Net with Multiple Rounds

4.5 Efficiency

On Figure 3, one can see the full construction of the ballots and their mixing using our probabilist signature (the other instantiation is proposed on Figure 5 in the Appentix B): one can note that a ballot $\mathcal{B}_i = (C_i, \ell_i, \sigma_i, \mathsf{vk}_i, \Sigma_i) \in \mathbb{G}_1^5 \times \mathbb{G}_2^7 \times \mathbb{G}_1$, globally contains 6 elements from $\mathbb{G}_1$ and 7 elements from $\mathbb{G}_2$. This is a bit more than the optimal ballot that would just contain the ciphertext (2 elements from $\mathbb{G}_1$), the signature, and the certified key (which all together should essentially correspond to 5 group elements in size), but not that much. In addition, there are the common elements $\overline{C}_0$ and $\overline{\mathsf{vk}}_r$, that are defined by $(\ell, g, h) \in \mathbb{G}_1^3$ and $\mathfrak{g}_r \in \mathbb{G}_2$, and $\mathsf{VK} = (g_j)_j \in \mathbb{G}_1^5$, where $\mathsf{EK} = h \in \mathbb{G}_1$.

Hence, the ballot-box contains: 7 elements from $\mathbb{G}_1$, for the system parameters, and for each round, 1 element from $\mathbb{G}_2$, plus n times 6 elements from $\mathbb{G}_1$ and 7 elements from $\mathbb{G}_2$.

After one mixing, one outputs new round parameter (1 element from $\mathbb{G}_2$) and n times 6 elements from $\mathbb{G}_1$ and 7 elements from $\mathbb{G}_2$, plus the Diffie-Hellman proof: 2 elements from $\mathbb{G}_1$ and 2 elements from $\mathbb{G}_2$. Even after multiple mixing steps, the amount of data will not grow.

For a mixing, the prover only needs to make 7 exponentiations in $\mathbb{G}_1$ and 9 exponentiations in $\mathbb{G}_2$ per ciphertext plus an overhead of 1 exponentiation in $\mathbb{G}_2$, for the round parameter.

After one mixing or more, the verifier only needs to make n times 9 pairings to verify σ_i, 10 pairings to verify Σ_i plus, just once, 12 pairings to verify the global proof of Diffie-Hellman. One may note that the verification time does not increase with the number of mixes which is the most important property of this mix-net: $12 + 19n$ pairing evaluations for the verification, whatever the number of rounds.

5 Security Analysis

Let us now formally prove the security properties, which should be, as described on Figure 4:

- soundness: the output ballot-box contains a permutation of randomizations of the input ballot-box (for honest voters)
- privacy: one cannot link an input ciphertext to an output ciphertext.

In this part, we focus on proving these security notions with our *efficient* non-miscible linearly-homorphic signature (see Section 3.5). Actually, the soundness does not depend on the specific signature used for Σ_i, but just on non-miscibility. But, the unlinkability will be proven just for this specific signature (see the Appendix B for the scheme and proof of unlinkability with just a linarly-homormorphic signature for Σ_i and Square Diffie-Hellman Tuples for the non-miscibility property).

Parameters

For a security parameter κ, let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, \mathfrak{g}, e) \leftarrow \mathcal{G}(\kappa)$, where g and $\mathfrak{g}$ are random generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively, and $(v_{1,1}, v_{1,2}, v_{2,1}, v_{2,2})$ a Diffie-Hellman tuple, then the global parameters are $\mathsf{Gparam} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, \mathfrak{g}, e, v_{1,1}, v_{1,2}, v_{2,1}, v_{2,2})$;
Let the system keys be $\mathsf{SK} = (S_1, S_2, S_3, S_4, S_5) \xleftarrow{\$} \mathbb{Z}_p^5$ the signing key, $\mathsf{VK} = (g_j = g^{S_j})_j$ the associated verification key, $\mathsf{DK} = d \xleftarrow{\$} \mathbb{Z}_p$ the private key and $\mathsf{EK} = h = g^d$ the corresponding encryption key. It also chooses $\ell \xleftarrow{\$} \mathbb{G}_1$.
The round parameter is initialized as $\mathfrak{g}_r \leftarrow \mathfrak{g}$.

Initialization $(r = 0)$

- Keys: $\mathsf{sk}_i = (u_i, v_i, x_i, y_i) \xleftarrow{\$} \mathbb{Z}_p^4$, $\mathsf{vk}_i = (\mathfrak{f}_i = \mathfrak{g}_r^{u_i}, \mathfrak{l}_i = \mathfrak{g}_r^{v_i}, \mathfrak{g}_i = \mathfrak{g}_r^{x_i}, \mathfrak{h}_i = \mathfrak{g}_r^{y_i})$;
  $\Sigma_i = \left(\Sigma_{i,1} = (\mathfrak{g}_r^{S_1} \mathfrak{f}_i^{S_2} \mathfrak{l}_i^{S_3} \mathfrak{g}_i^{S_4} \mathfrak{h}_i^{S_5})^{1/R_i}, \Sigma_{i,2} = \mathfrak{g}_r^{S_3/R_i}, T_{i,1} = \mathfrak{g}^{R_i}, T_{i,2} = g^{R_i}\right)$, for $R_i \xleftarrow{\$} \mathbb{Z}_p$, from the system authority. One can randomize the signature with $R'_i \xleftarrow{\$} \mathbb{Z}_p$: $\Sigma_i = \left(\Sigma_{i,1}^{1/R'_i}, \Sigma_{i,2}^{1/R'_i}, T_{i,1}^{R'_i}, T_{i,2}^{R'_i}\right)$
- Ballot $\mathcal{B}_i$: $C_i = (a_i = g^{r_i}, b_i = h^{r_i} M_i)$, for $r_i \xleftarrow{\$} \mathbb{Z}_p$, $C_0 = (g, h)$, expanded into $\overline{C}_i = (g, \ell_i, a_i, b_i)$ and $\overline{C}_0 = (1, \ell, g, h)$, for $\ell_i \xleftarrow{\$} \mathbb{G}_1$, signed into $\sigma_i = (\sigma_{i,1} = g^{u_i} \ell_i^{v_i} a_i^{x_i} b_i^{y_i}, \sigma_{i,2} = \ell^{v_i} g^{x_i} h^{y_i})$.
  $\mathcal{B}_i = (C_i, \ell_i, \sigma_i, \mathsf{vk}_i, \Sigma_i)$ and the round parameter $\mathfrak{g}_r$ defines $\overline{\mathsf{vk}}_r = (1, 1, \mathfrak{g}_r, 1, 1)$.

Mixing

From an input ballot-box containing $\mathfrak{g}_r$, C_0, and $(C_i, \ell_i, \sigma_i, \mathsf{vk}_i, \Sigma_i)_i$ for all the voters with the global proof $(\mathsf{Com} = (c, d), \Theta, \Psi)$, construct the output ballot-box $\mathfrak{g}'_r$, $(C'_i, \ell'_i, \sigma'_i, \mathsf{vk}'_i, \Sigma'_i)_i$ where $\alpha \xleftarrow{\$} \mathbb{Z}_p$: $\mathfrak{g}'_r = \mathfrak{g}_r^{\alpha}$, and for each ballot i, $\gamma_i, \delta_i, \mu_i \xleftarrow{\$} \mathbb{Z}_p$:

$$a'_i = a_i \cdot g^{\gamma_i} \qquad b'_i = b_i \cdot h^{\gamma_i} \qquad \ell'_i = \ell_i \cdot \ell^{\gamma_i}$$
$$\sigma'_{i,1} = \sigma_{i,1} \cdot \sigma_{i,2}^{\gamma_i} \cdot \ell'^{\delta_i}_i \qquad \sigma'_{i,2} = \sigma_{i,2} \cdot \ell'^{\delta_i}_r$$
$$\mathfrak{f}'_i = \mathfrak{f}_i^{\alpha} \qquad \mathfrak{l}'_i = (\mathfrak{l}_i \cdot \mathfrak{g}_r^{\delta_i})^{\alpha} \qquad \mathfrak{g}'_i = \mathfrak{g}_i^{\alpha} \qquad \mathfrak{h}'_i = \mathfrak{h}_i^{\alpha}$$
$$\Sigma'_{i,1} = (\Sigma_{i,1} \cdot \Sigma_{i,2}{}^{\delta_i})^{\alpha/\mu_i} \qquad \Sigma'_{i,2} = \Sigma_{i,2}^{\alpha/\mu_i} \qquad T'_{i,1} = T_{i,1}^{\mu_i} \qquad T'_{i,2} = T_{i,2}^{\mu_i}$$

Output all the tuples $(C'_i, \ell'_i, \sigma'_i, \mathsf{vk}'_i, \Sigma'_i)_i$ in random order with $\mathfrak{g}'_r$ and the global proof π' of $(\mathfrak{g}_r, \mathfrak{g}'_r, \prod \mathfrak{f}_i, \prod \mathfrak{f}'_j)$ is a Diffie-Hellman tuple (as detailed in the figure 2).

Verification

The verifier will check this proof π', the number of input tuples is the same as the number of output tuples, the $\mathfrak{f}'_i$ are all distinct, and the signatures σ'_i and Σ'_i are valid on individual output tuples.

Figure 3. Detailed Shuffling of ElGamal Ciphertexts

5.1 Soundness: Proof of Permutation

First, we prove the soundness of the mixing: given an input ballot-box $\mathcal{BB}\mathsf{ox} = \{\mathcal{B}_i, i \in \mathcal{I}\}$, where ballot $\mathcal{B}_i$ encrypts vote v_i, and an output ballot-box $\mathcal{BB}\mathsf{ox}' = \{\mathcal{B}'_j, j \in \mathcal{J}\}$, where ballot $\mathcal{B}'_j$ encrypts vote v'_j, there exists a bijection Π from $\mathcal{I}$ into $\mathcal{J}$ such that $v_i = v'_{\Pi(i)}$, for all $i \in \mathcal{I}$. At least, we can prove that, for all the ballots of the honest voters, $\mathcal{I}' \subseteq \mathcal{I}$, there is a bijection Π from $\mathcal{I}$ into $\mathcal{J}$ such that $v_i = v'_{\Pi(i)}$ for all $i \in \mathcal{I}'$. This means that a dishonest server, with the help of dishonest (but legitimate) voters, could replace some votes (of these dishonest voters) by some other votes (from dishonest voters), but votes of honest voters cannot be altered, which is the most important, as there is no reason to provide any guarantee to dishonest users.

To achieve this security notion, we need the voters to prove the knowledge of their signing keys sk_i (or at least u_i) to get their verification key vk_i signed into Σ_i. Let us assume this is performed using a zero-knowledge proof of knowledge that admits a straightline extractor.

Soundness (for Honest Voters)

Given an input ballot-box $\mathcal{BB}\mathsf{ox} = \{\mathcal{B}_i, i \in \mathcal{I}\}$, where ballot $\mathcal{B}_i$ encrypts vote V_i, and an output ballot-box $\mathcal{BB}\mathsf{ox}' = \{\mathcal{B}'_j, j \in \mathcal{J}\}$, generated by an adversary $\mathcal{A}$, where ballot $\mathcal{B}'_j$ encrypts vote V'_j, if all the verification checks pass, there should exist a bijection Π from $\mathcal{I}$ into $\mathcal{J}$ such that $V_i = V'_{\Pi(i)}$, for all honest i in $\mathcal{I}$, with overwhelming probability.

Privacy

Given an input ballot-box $\mathcal{BB}\mathsf{ox} = \{\mathcal{B}_i, i \in \mathcal{I}\}$ and a mixed output ballot-box $\mathcal{BB}\mathsf{ox}' = \{\mathcal{B}'_j, j \in \mathcal{J}\}$, for a random bijection Π from $\mathcal{I}$ into $\mathcal{J}$ such that $\mathcal{B}_i$ is randomized in $\mathcal{B}'_{\Pi(i)}$ for all $i \in \mathcal{I}$, for any pair of honestly generated ballots $\mathcal{B}_{i_0}$ and $\mathcal{B}_{i_1}$, if one provides (i_0, i_1, j_0, j_1), where $j_b = \Pi(i_0)$ and $j_{1-b} = \Pi(i_1)$, for a random bit b, any adversary $\mathcal{A}$ should have a negligible advantage in guessing b.

Figure 4. Mixing Security Properties

Non-Miscibility of the Verification Keys. The first step is the guarantee that every output verification key vk'_j corresponds to a legitimate verification key vk_i, that has been initially signed by the authority. From the legitimate (expanded) verification keys $(\overline{\mathsf{vk}}_i = (\mathfrak{g}_r, \mathfrak{f}_i, \mathfrak{l}_i, \mathfrak{g}_i, \mathfrak{h}_i))_i$ and $\overline{\mathsf{vk}}_r = (1, 1, \mathfrak{g}_r, 1, 1)$, signed under VK into $(\Sigma_i)_i$, with a signature scheme $(\mathsf{Keygen}^*, \mathsf{Sign}^*, \mathsf{Verif}^*)$ that is linearly-homomorphic and guarantees non-miscibility, any mix-server must output a new (expanded) verification key $\overline{\mathsf{vk}}'_j$ with a new signature Σ'_j. The non-miscibility of the signature scheme $(\mathsf{Keygen}^*, \mathsf{Sign}^*, \mathsf{Verif}^*)$ proves the following proposition:

Proposition 12 (Legitimate Output). *Under the non-miscibility and the unforgeability of the signature scheme* $(\mathsf{Keygen}^*, \mathsf{Sign}^*, \mathsf{Verif}^*)$*, for any output ballot with (expanded) verification key* $\overline{\mathsf{vk}}'_j$ *there exists a related legitimate verification key* $\overline{\mathsf{vk}}_i$ *such that* $\overline{\mathsf{vk}}'_j = \overline{\mathsf{vk}}_i^{\alpha} \times \overline{\mathsf{vk}}_r^{z_i}$*, for some scalar* z_i*, and* α *such that* $\mathfrak{g}'_r = \mathfrak{g}_r^{\alpha}$*.*

We recall that $\Sigma_i = (\Sigma_{i,1} = (\mathfrak{g}_r^{S_1}\mathfrak{f}_i^{S_2}\mathfrak{l}_i^{S_3}\mathfrak{g}_i^{S_4}\mathfrak{h}_i^{S_5})^{1/R_i}, \Sigma_{i,2} = \mathfrak{g}_r^{S_3/R_i}, T_{i,1} = \mathfrak{g}^{R_i}, T_{i,2} = g^{R_i})$, for $R_i \xleftarrow{\$} \mathbb{Z}_p$, is the combined signature of $\overline{\mathsf{vk}}_i = (\mathfrak{g}_r, \mathfrak{f}_i, \mathfrak{l}_i, \mathfrak{g}_i, \mathfrak{h}_i)$ and $\overline{\mathsf{vk}}_r = (1, 1, \mathfrak{g}_r, 1, 1)$. We use this signature with a fixed system verification key VK. Thus, the only operations are among $\overline{\mathsf{vk}}_i$ and $\overline{\mathsf{vk}}_r$: $\overline{\mathsf{vk}}'_i = (\overline{\mathsf{vk}}_i \times \overline{\mathsf{vk}}_r^{\delta_i})^{\alpha}$ and $\overline{\mathsf{vk}}'_r = \overline{\mathsf{vk}}_r^{\alpha}$. This just impacts $\Sigma'_{i,1} = (\Sigma_{i,1} \cdot \Sigma_{i,2}^{\delta_i})^{\alpha}$ and $\Sigma'_{i,2} = \Sigma_{i,2}^{\alpha}$. We eventually randomize R_i with μ_i: Σ'_i becomes a signature of the combined $\overline{\mathsf{vk}}'_i = (\mathfrak{g}'_r, \mathfrak{f}'_i, \mathfrak{l}'_i, \mathfrak{g}'_i, \mathfrak{h}'_i)$ and $\overline{\mathsf{vk}}'_r = (1, 1, \mathfrak{g}'_r, 1, 1)$.

However it does not exclude the server to insert a ballot with a legitimate verification key vk_i, which was not in the initial ballot-box. Nevertheless note that no voter can have two ballots, because of the collision checks on the output $\mathfrak{f}'_j$'s, if there was no collision on the input $\mathfrak{f}_i$'s, as the same α is enforced for all the randomizations.

Permutation of the Verification Keys. Hence, the second step consists in proving that any honest input verification key corresponds to an output verification key. Note that the mix-server will add the proof that $(\mathfrak{g}_r, \mathfrak{g}'_r, \prod_i \mathfrak{f}_i, \prod_j \mathfrak{f}'_j)$ is a Diffie-Hellman tuple where the products are on the ballots in the initial ballot-box for the $\mathfrak{f}_i$'s and in the output ballot-box for the $\mathfrak{f}'_j$'s: $\sum_i u_i$ are the same on the input ballots and the output ballots.

As there are no duplicates (check of collisions on the $\mathfrak{f}_i$ and $\mathfrak{f}'_j$) and the numbers of input ballots and output ballots are the same, a malicious mix-server must replace an input ballot by a new one: if $\mathcal{N}$ is the set of new ballots and $\mathcal{D}$ the set of deleted ballots, $\prod_{\mathcal{D}} \mathfrak{f}_i = \prod_{\mathcal{N}} \mathfrak{f}_j$.

Let us be given a tuple $(\mathfrak{g}, \mathfrak{f} = \mathfrak{g}^u, g, f = g^u)$, as input of a TDL challenge in $\mathbb{G}_1$ and $\mathbb{G}_2$: the simulator will guess the honest user i^* that will be deleted, and implicitly sets $u_{i^*} = u$, with $\mathfrak{f}_{i^*}$, which allows it to use $f = g^{u_{i^*}}$ in the signature of $\overline{C}_{i^*}$ on the first component g, while all the other scalars are chosen by the simulator $(v_{i^*}, x_{i^*}, y_{i^*})$, as well as the authority signing keys, and, for all the other users, the secret keys $\mathsf{sk}_i = (u_i, v_i, x_i, y_i)$ can be extracted at the certification time (using the straightline extractor from the zero-knowledge proof of knowledge). Note that the zero-knowledge simulator is used for i^*.

If some honest user is deleted in the output ballot-box, with probability greater than $1/n$, this is i^*: but as proven above, $\sum_{\mathcal{D}} u_i = \sum_{\mathcal{N}} u_j$, and so $u_{i^*} = \sum_{\mathcal{N}} u_j - \sum_{\mathcal{D}\setminus\{i^*\}} u_i$, which breaks the twin discrete logarithm assumption. This proves the following proposition:

Proposition 13 (Permutation of Keys). *Given a set of legitimate verification keys vk_i's with signing keys known to the honest users only, from any input ballot-box (with verification keys vk_i, for $i \in \mathcal{I}$) and transformed ballot-box (with verification key vk'_j, for $j \in \mathcal{J}$) such that all the $\mathfrak{f}_i$'s are distinct, all the $\mathfrak{f}'_j$'s are distinct, $|\mathcal{I}| = |\mathcal{J}|$, and $(\mathfrak{g}_r, \mathfrak{g}'_r, \prod_i \mathfrak{f}_i, \prod_j \mathfrak{f}'_j)$ is a Diffie-Hellman tuple, then there exists α and a bijection Π from $\mathcal{I}$ into $\mathcal{J}$ such that for any honest i, $\mathsf{vk}'_{\Pi(i)} = (\mathsf{vk}_i \times \mathsf{vk}_r^{\delta_i})^\alpha$ for some scalar δ_i, under the unforgeability of the signature schemes and the TDL assumption.*

We stress that for this property to hold, the signing keys (at least the u_i's) of the honest players must be random and unknown to the malicious mix-server. We will thus assume the signing keys are generated by the users themselves and the verification keys signed by the authority (after a proof of knowledge of the signing keys).

Permutation of Plaintexts. Eventually, the last step consists in proving that output ciphertexts correspond to randomizations of permuted input ciphertexts. We have just proven that this is true for the verification keys: there exists a bijection Π from $\mathcal{I}$ into $\mathcal{J}$ such that $\mathsf{vk}'_{\Pi(i)} = (\mathsf{vk}_i \times \mathsf{vk}_r^{\delta_i})^\alpha$ for some scalar δ_i, for all the honest voters i among the input voters in $\mathcal{I}$.

From the signature verification on the output tuples, $C'_{\Pi(i)}$ is signed under $\mathsf{vk}'_{\Pi(i)}$ in $\sigma'_{\Pi(i),1}$, for every i: $e(\sigma'_{\Pi(i),1}, \mathfrak{g}'_r) = e(g, \mathfrak{f}_i^\alpha) \cdot e(\ell'_{\Pi(i)}, \mathfrak{l}_i^\alpha \mathfrak{g}_r^{\alpha\delta_i}) \cdot e(a'_{\Pi(i)}, \mathfrak{g}_i^\alpha) \cdot e(b'_{\Pi(i)}, \mathfrak{h}_i^\alpha)$, and since the same α appears in $\mathfrak{g}'_r = \mathfrak{g}_r^\alpha$, then for every i, we have

$$\begin{aligned} e(\sigma'_{\Pi(i)}, \mathfrak{g}_r) &= e(g, \mathfrak{f}_i) \cdot e(\ell'_{\Pi(i)}, \mathfrak{l}_i \mathfrak{g}_r^{\delta_i}) \cdot e(a'_{\Pi(i)}, \mathfrak{g}_i) \cdot e(b'_{\Pi(i)}, \mathfrak{h}_i) \\ &= e(g, \mathfrak{f}_i) \cdot e(\ell'_{\Pi(i)}, \mathfrak{l}_i) \cdot e(a'_{\Pi(i)}, \mathfrak{g}_i) \cdot e(b'_{\Pi(i)}, \mathfrak{h}_i) \cdot e({\ell'}^{\delta_i}_{\Pi(i)}, \mathfrak{g}_r) \end{aligned}$$

and so $\sigma'_{\Pi(i)} / {\ell'}^{\delta_i}_{\Pi(i)}$ is a signature of $\overline{C}'_{\Pi(i)} = (g, \ell'_{\Pi(i)}, a'_{\Pi(i)}, b'_{\Pi(i)})$ under vk_i: under the unforgeability assumption of the signature scheme, $C'_{\Pi(i)}$ is necessarily a linear combination of the already signed vectors under vk_i, which are C_i and C_0, with some coefficients u, v: $a'_{\Pi(i)} = a_i^u g_r^v$, $b'_{\Pi(i)} = b_i^u h_r^v$, and $g = g^u 1^v$. Hence, $u = 1$, which means that $C'_{\Pi(i)}$ is a randomization of C_i.

Proposition 14 (Permutation of Ciphertexts). *Given a set of legitimate verification keys vk_i's with signing keys known to the users only, from any input ballot-box (with verification keys vk_i, for $i \in \mathcal{I}$) and transformed ballot-box (with verification key vk'_j, for $j \in \mathcal{J}$) such that all the*

$\mathfrak{f}_i$*'s are distinct, all the* $\mathfrak{f}'_j$*'s are distinct,* $|\mathcal{I}| = |\mathcal{J}|$*, and* $(\mathfrak{g}_r, \mathfrak{g}'_r, \prod_i \mathfrak{f}_i, \prod_j \mathfrak{f}'_j)$ *is a Diffie-Hellman tuple, then there exists a bijection* Π *from* $\mathcal{I}$ *into* $\mathcal{J}$ *such that for any honest voter* i*,* $C'_{\Pi(i)}$ *is a randomization of* C_i*. This statement holds under the unforgeability of the signature schemes and the* TDL *assumption.*

We stress that this proposition only guarantees permutation of ciphertexts for honest voters. There is indeed no formal guarantee for compromised voters who would have revealed their signing keys, as the mix-server could

- replace the ciphertexts of some compromised voters, and sign them with their revealed signing keys;
- or find two sets $\mathcal{D}$ and $\mathcal{N}$ such that $\sum_{\mathcal{D}} u_i = \sum_{\mathcal{N}} u_j$, and then delete the votes from the users in $\mathcal{D}$ and insert new votes for the users in $\mathcal{N}$, using the revealed signing keys to sign the ciphertexts;
- or even ask the voters to generate related signing keys, to make easy to find such sets $\mathcal{D}$ and $\mathcal{N}$ such that $\sum_{\mathcal{D}} u_i = \sum_{\mathcal{N}} u_j$.

As a consequence, our result that guarantees a permutation on the honest ballots is optimal. We cannot guarantee anything for the compromised voters: to limit compromise, the voters should destroy their signing keys after use. An alternative would be to generate the signing keys in a distributed way, between the voters and the authority, so that nobody knows the signing keys. But then a distributed signing process would be required. This is detailed in Section 6.3.

5.2 Privacy: Unlinkability

After proving the soundness, we have to prove the anonymity (a.k.a. unlinkability), which can also be seen as zero-knowledge property with respect to outsider verifiers. To this aim, we will show that from a ballot-box, if the simulator randomizes the ballots $\mathcal{B}_{i_0}$ and $\mathcal{B}_{i_1}$ into $\mathcal{B}'_{j_b}$ and $\mathcal{B}'_{j_{1-b}}$, no adversary can guess b.

We stress that in the following proof, we assume (at least) two input ballots to have been generated by honest (non-compromised) voters. More precisely, at least two signing keys should not be known to the adversary, nor the random coins used by the mix-server. We start from the initial real game that generates the view of the adversary, for the input ballots and the randomized ballots, and conclude with a game where the randomized ballots are all generated randomly, independently of the input ballots: b is thus perfectly hidden.

Game $\mathbf{G}_0$: One first generates a group structure $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, \mathfrak{g}, e) \leftarrow \mathcal{G}(\kappa)$, and a Diffie-Hellman tuple $(v_{1,1}, v_{1,2}, v_{2,1}, v_{2,2})$ in $\mathbb{G}_1$. One then sets the global parameters, $\mathsf{Gparam} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, \mathfrak{g}, e, v_{1,1}, v_{1,2}, v_{2,1}, v_{2,2})$. One generates SK and VK for the authority signature, and randomly chooses $d \xleftarrow{\$} \mathbb{Z}_p$ to generate the server public key $\mathsf{EK} = h = g^d$, and the noise parameter $\ell \xleftarrow{\$} \mathbb{G}_1$. One also sets $\mathsf{vk}_r = (1, \mathfrak{g}_r = \mathfrak{g}^A, 1, 1)$ and $C_0 = \mathsf{Encrypt}_{\mathsf{EK}}(1) = (g, h)$. One can expand them to $\overline{\mathsf{vk}}_r = (1, \mathsf{vk}_r)$ and $\overline{C}_0 = (1, \ell, C_0)$. Actually, $A = 1$ in the initial step, when the user encrypts his message M_i, but since the shuffling may happens after several other shuffling iterations, we have the successive exponentiations to multiple α (in A) for vk_r.

For each honest user i, one randomly chooses $u_i, v_i, x_i, y_i, r_i, \rho_i \xleftarrow{\$} \mathbb{Z}_p$ to generate $\mathsf{vk}_i = (\mathfrak{f}_i = \mathfrak{g}_r^{u_i}, \mathfrak{l}_i = \mathfrak{g}_r^{v_i}, \mathfrak{g}_i = \mathfrak{g}_r^{x_i}, \mathfrak{h}_i = \mathfrak{g}_r^{y_i})$, $\overline{\mathsf{vk}}_i = (\mathfrak{g}_r, \mathsf{vk}_i)$, and the signature Σ_i of $(\overline{\mathsf{vk}}_i, \overline{\mathsf{vk}}_r)$ under SK with respect to g, as well as $C_i = \mathsf{Encrypt}_{\mathsf{EK}}(M_i) = (a_i = g^{r_i}, b_i = h^{r_i} M_i)$, $\overline{C}_i = (g, \ell_i = \ell^{\rho_i}, C_i)$, and the signature σ_i of $(\overline{C}_i, \overline{C}_0)$ under sk_i with respect to $\mathfrak{g}_r$. Indeed, knowing the signing keys allows to honestly generate the signatures Σ_i and σ_i. $\mathcal{B}_i = (\overline{C}_i, \sigma_i, \mathsf{vk}_i, \Sigma_i)$ constitutes a ballot of an honest user. For the corrupted users, the simulator directly receives $\mathcal{B}_i = (\overline{C}_i, \sigma_i, \mathsf{vk}_i, \Sigma_i)$. The input ballot-box is then $\mathcal{BB}\mathsf{ox} = \{\mathcal{B}_i\}_i \cup \{(\mathsf{vk}_r, \overline{C}_r)\}$ including the ballots of all the honest and corrupted users and the round parameters.

By randomly choosing global $\alpha \xleftarrow{\$} \mathbb{Z}_p$ and individual $\gamma_i, \delta_i, \mu_i \xleftarrow{\$} \mathbb{Z}_p$ for all the users, the simulator also generates, as the mix-server would do, $\mathfrak{g}'_r = \mathfrak{g}_r^\alpha$, to update

$$\mathsf{vk}'_i = (\mathfrak{f}'_i = \mathfrak{f}_i^\alpha, \mathfrak{l}'_i = (\mathfrak{l}_i \cdot \mathfrak{g}_r^{\delta_i})^\alpha, \mathfrak{g}'_i = \mathfrak{g}_i^\alpha, \mathfrak{h}'_i = \mathfrak{h}_i^\alpha) \qquad \overline{\mathsf{vk}}'_i = (\mathfrak{g}'_r, \mathsf{vk}'_i) = (\overline{\mathsf{vk}}_i \cdot \overline{\mathsf{vk}}_r^{\delta_i})^\alpha$$
$$\mathsf{vk}'_r = (1, \mathfrak{g}'_r, 1, 1) \qquad \overline{\mathsf{vk}}'_r = (1, \mathsf{vk}'_r) = \overline{\mathsf{vk}}_r^\alpha$$

$$C'_i = (a'_i = a_i \cdot g_r^{\gamma_i}, b'_i = b_i \cdot h_r^{\gamma_i}) \qquad \overline{C}'_i = (1, \ell'_i = \ell_i \cdot \ell_r^{\gamma_i}, C'_i) = \overline{C}_i \cdot \overline{C}_0{}^{\gamma_i}$$
$$\sigma'_{i,1} = \sigma_{i,1} \cdot \sigma_{i,2}^{\gamma_i} \cdot \ell'_i{}^{\delta_i} \qquad \sigma'_{i,2} = \sigma_{i,2} \cdot \ell'_r{}^{\delta_i}$$

$$\Sigma'_{i,1} = (\Sigma_{i,1} \cdot \Sigma_{i,2}{}^{\delta_i})^{\alpha/\mu_i} \qquad \Sigma'_{i,2} = \Sigma_{i,2}^{\alpha/\mu_i} \qquad T'_{i,1} = T_{i,1}^{\mu_i} \qquad T'_{i,2} = T_{i,2}^{\mu_i}$$

Eventually, the simulator chooses a random permutation Π on $\{1, \ldots, n\}$, and sets the output ballots $\mathcal{B}'_i = (\overline{C}'_{\Pi(i)}, \sigma'_{\Pi(i)}, \mathsf{vk}'_{\Pi(i)}, \Sigma'_{\Pi(i)})$. It outputs $\mathcal{BB}\mathsf{ox} = \{\mathcal{B}_i\}_i \cup \{(\mathsf{vk}_r, \overline{C}_0)\}$ and $\mathcal{BB}\mathsf{ox}' = \{\mathcal{B}'_i\} \cup \{(\mathsf{vk}'_r, \overline{C}_0)\}$, together with a zero-knowledge proof π' of Diffie-Hellman tuple for $(\mathfrak{g}_r, \mathfrak{g}'_r, \prod \mathfrak{f}_i, \prod \mathfrak{f}'_j)$.

Game $\mathbf{G}_1$: Everything is the same, except the proof π' of Diffie-Hellman tuple for $(\mathfrak{g}_r, \mathfrak{g}'_r, \prod \mathfrak{f}_i, \prod \mathfrak{f}'_j)$, that is generated using the simulator that does not need to know α (from the zero-knowledge property).

Game $\mathbf{G}_2$: For all the honest users, we replace C_i by an encryption of 1: $C_i = \mathsf{Encrypt}_{\mathsf{EK}}(1) = (a_i = g^{r_i}, b_i = h^{r_i})$ and replace the random value $\ell_i = \ell^{\rho_i}$ by $\ell_i = \ell^{r_i}$. Hence, we set $\overline{C}_i = (g, \ell_i = \ell^{r_i}, a_i = g^{r_i}, b_i = h^{r_i})$ for $r_i \xleftarrow{\$} \mathbb{Z}_p$.

Lemma 15. *Under the* DDH *assumption in $\mathbb{G}_1$, this game is computationally indistinguishable from the previous one.*

Game $\mathbf{G}_3$: For all the honest users, we still choose at random the signing keys $\mathsf{sk}_i = (u_i, v_i, x_i, y_i)$ but we also choose at random $\mathsf{sk}'_i = (u'_i, v'_i, x'_i, y'_i)$ in order to have random verification keys vk'_i.

Lemma 16. *Under the* DDH *assumption in $\mathbb{G}_2$ and Unlinkability Assumption (see Definition 6) this game is computationally indistinguishable from the previous one.*

Since in the final game, the verification keys vk'_i are truly random, and independent from vk_i, and $\overline{C}'_i$ contains random independent elements:

Proposition 17 (Unlinkability of the Mixing). *Given any input ballot-box and transformed ballot-box (following the protocol from Figure 3), for any pair of honestly generated input ballots $(\mathcal{B}_0, \mathcal{B}_1)$ and pair of output ballots $(\mathcal{B}'_0, \mathcal{B}'_1)$, such that $\mathcal{B}'_b$ is a randomization of $\mathcal{B}_0$, and $\mathcal{B}'_{1-b}$ is a randomization of $\mathcal{B}_1$, for a random bit $b \xleftarrow{\$} \{0, 1\}$, no adversary can get a non-negligible advantage in guessing b.*

We stress that for this property to hold, the adversary should not know the two secret keys associated to the input ballots. That means the indistinguishability is for outsider adversaries only (none of the two target voters who generated $\mathcal{B}_0$ and $\mathcal{B}_1$), that do not either know the random coins of the mix-server.

One also has to make sure that Σ'_i and Σ_i have no invariant that can be detected: if they are deterministic (as our basic linearly-homomorphic scheme with Square Diffie-Hellman tuples), there is no problem, if they are probabilistic, they must be randomizable (as our scheme that guarantees non-miscibility).

We can show again that our security against outsider adversaries is optimal, as the owner of $\mathcal{B}_0$ knows u_0, and can check whether $\mathfrak{f}'_0 = \mathfrak{g}'_r{}^{u_0}$ or not. But all the other users have no way to guess b.

Proof (Lemma 15). The proof works with a sequence of hybrid games:

Game $\mathbf{G}_0$: Since the decryption key d is never used, we are just given g and $\mathsf{EK} = h$ and by an hybrid game for each honest user i, we can replace C_i by an encryption of 1: $C_i = \mathsf{Encrypt}_{\mathsf{EK}}(1) = (a_i = g^{r_i}, b_i = h^{r_i})$. Under the DDH in $\mathbb{G}_1$ assumption this game is indistinguishable from the previous one.

Game $\mathbf{G}_1$: We now know again d, and since ℓ is random, by an hybrid game for each honest user i, we can replace the random value $\ell_i = \ell^{\rho_i}$ by a Diffie-Hellman value with $(g, a_i = g^{r_i}, \ell)$, and thus $\ell_i = \ell^{r_i}$, for a random unknown r_i, and complete: $C_i = \mathsf{Encrypt}_{\mathsf{EK}}(1) = (a_i = g^{r_i}, b_i = a_i^d)$ and then $\overline{C}_i = (g, \ell_i = \ell^{r_i}, C_i)$. Under the DDH in $\mathbb{G}_1$ assumption this game is indistinguishable from the previous one. □

Proof (Lemma 16). Before entering into the sequence of games for this lemma, in order to randomize honest ballots, let us explain how we will generate the randomized ballots for corrupted users: as we assumed the signatures Σ_i provided by the authority with the view of the proof of knowledge of sk_i, our simulator has access to $\mathsf{sk}_i = (u_i, v_i, x_i, z_i)$ for all the corrupted users. The mixing step consists in updating the ciphertexts, the keys and the signatures, and we show how to do it without knowing α, but just (g, h, ℓ) and the round parameter $\mathfrak{g}_r$, as well as the individual random coins γ_i and δ_i: first, $\overline{C}'_i = \overline{C}_i \cdot \overline{C}_0^{\gamma_i}$, and $\mathsf{vk}'_i = (\mathsf{vk}_i \cdot {\mathsf{vk}_r}^{\delta_i})^{\alpha}$, but with respect to $\mathfrak{g}'_r = \mathfrak{g}_r$, hence $\mathsf{sk}'_i = (u_i, v_i + \delta_i, x_i, y_i)$, which makes easy the computation of vk'_i from $\mathfrak{g}'_r$, as well as the signature σ'_i of any message, using sk'_i. Eventually, Σ'_i can also be generated using the authority signing key SK, that we know, and a random R'_i.

In the sequence of games below, we will thus only show how to generate round parameter and honest input/output ballots, as the ballots for corrupted users will be simulated as above. We now run an hybrid game, for $1 \le i \le n'$, where n' is the number of honest users. For all the honest users, the simulator randomly chooses the signing keys sk_j: for all $j \ge i$, one chooses $\gamma_i, \delta_i \xleftarrow{\$} \mathbb{Z}_p$, and does the simulation of $\mathcal{B}'_j$ as above, for the corrupted users, in order to generate sk'_j, $\overline{C}'_j$ and vk'_j, as well as σ'_j; for all $j < i$, one just chooses $\gamma_i \xleftarrow{\$} \mathbb{Z}_p$ in order to generate $\overline{C}'_j$, but $\mathsf{sk}'_j \xleftarrow{\$} \mathbb{G}_1^4$ to generate vk'_j and σ'_j. Note that SK is known to generate Σ'_j.

Of course, when $i = 1$, this first game is exactly the original game, where all the honest randomizations are performed correctly.

Game $\mathbf{G}_0$: In this game, we still choose a random $d \xleftarrow{\$} \mathbb{Z}_p$ for $h = g^d$, but also a random $e \xleftarrow{\$} \mathbb{Z}_p$ for $\ell = g^e$. Then we can simulate

$$\overline{C}_i = (g, \ell_i = \ell^{r_i}, a_i = g^{r_i}, b_i = h^{r_i}) \qquad \overline{C}'_i = (g, \ell'_i = \ell^{r'_i}, a'_i = g^{r'_i}, b'_i = h^{r'_i})$$

for known random scalars $r_i, r'_i \xleftarrow{\$} \mathbb{Z}_p$, where r'_i is actually $r_i + \gamma_i$. The signatures (σ_i, σ'_i) and (Σ_i, Σ'_i) are still simulated using the signing keys. The former satisfy:

$$\begin{aligned}
e(\sigma_{i,1}, \mathfrak{g}_r) &= e(g, \mathfrak{f}_i) \cdot e(\ell_i, \mathfrak{l}_i) \cdot e(a_i, \mathfrak{g}_i) \cdot e(b_i, \mathfrak{h}_i) = e(g, \mathfrak{f}_i(\mathfrak{g}_i \mathfrak{h}_i^d \mathfrak{l}_i^e)^{r_i}) \\
e(\sigma_{i,2}, \mathfrak{g}_r) &= e(\ell_r, \mathfrak{l}_i) \cdot e(g_r, \mathfrak{g}_i) \cdot e(h_r, \mathfrak{h}_i) = e(g_r, \mathfrak{g}_i \mathfrak{h}_i^d \mathfrak{l}_i^e) \\
e(\sigma'_{i,1}, \mathfrak{g}'_r) &= e(g, \mathfrak{f}'_i) \cdot e(\ell'_i, \mathfrak{l}'_i) \cdot e(a'_i, \mathfrak{g}'_i) \cdot e(b'_i, \mathfrak{h}'_i) = e(g, \mathfrak{f}'_i(\mathfrak{g}'_i \mathfrak{h}'^d_i \mathfrak{l}'^e_i)^{r'_i}) \\
e(\sigma'_{i,2}, \mathfrak{g}'_r) &= e(\ell'_r, \mathfrak{l}'_i) \cdot e(g'_r, \mathfrak{g}'_i) \cdot e(h'_r, \mathfrak{h}'_i) = e(g'_r, \mathfrak{g}'_i \mathfrak{h}'^d_i \mathfrak{l}'^e_i)
\end{aligned}$$

If we formally denote $\sigma_{i,1} = g^{s_i}$, $\sigma_{i,2} = g_r^{t_i}$, $\sigma'_{i,1} = g^{s'_i}$, $\sigma'_{i,2} = {g'_r}^{t'_i}$, then

$$\mathfrak{g}_r^{s_i} = \mathfrak{f}_i(\mathfrak{g}_i \mathfrak{h}_i^d \mathfrak{l}_i^e)^{r_i} \qquad \mathfrak{g}_r^{t_i} = \mathfrak{g}_i \mathfrak{h}_i^d \mathfrak{l}_i^e \qquad s_i = u_i + t_i r_i$$

and from

$$\begin{aligned}
{\mathfrak{g}_r}^{\alpha s'_i} = {\mathfrak{g}'_r}^{s'_i} = \mathfrak{f}'_i(\mathfrak{g}'_i \mathfrak{h}'^d_i \mathfrak{l}'^e_i)^{r'_i} = \mathfrak{f}_i^{\alpha}(\mathfrak{g}_i^{\alpha} \mathfrak{h}_i^{\alpha d}(\mathfrak{l}_i \mathfrak{g}_r^{\delta_i})^{\alpha e})^{r'_i} \\
{\mathfrak{g}_r}^{\alpha t'_i} = {\mathfrak{g}'_r}^{t'_i} = \mathfrak{g}'_i \mathfrak{h}'^d_i \mathfrak{l}'^e_i = {\mathfrak{g}_i}^{\alpha} {\mathfrak{h}_i}^{\alpha d}(\mathfrak{l}_i \mathfrak{g}_r^{\delta_i})^{\alpha e}
\end{aligned}$$

we also have

$$\mathfrak{g}_r{}^{s'_i} = \mathfrak{f}_i(\mathfrak{g}_i\mathfrak{h}_i^d\mathfrak{l}_i^e)^{r'_i}\mathfrak{g}_r^{e\delta_i r'_i} \qquad \mathfrak{g}_r{}^{t'_i} = (\mathfrak{g}_i\mathfrak{h}_i{}^d\mathfrak{l}_i^e)\mathfrak{g}_r^{\delta_i e} \qquad s'_i = u_i + t'_i r'_i$$

As consequence, $\sigma_{i,1} = g^{u_i} \cdot (g^{r_i})^{t_i} = g^{u_i} \cdot a_i^{t_i}$ and $\sigma'_{i,1} = g^{u_i} \cdot (g^{r'_i})^{t'_i} = g^{u_i} \cdot a_i'^{t'_i}$.

Game $\mathbf{G}_1$: Let us randomly choose scalars $u_i, r_i, r'_i, t_i, t'_i$ and α, then, from $(g, \mathfrak{g}_r)$, we can set $\mathfrak{g}'_r \leftarrow \mathfrak{g}_r^{\alpha}$, $a_i \leftarrow g^{r_i}$, $\sigma_{i,1} \leftarrow a_i^{t_i} g^{u_i}$, $\mathfrak{f}_i \leftarrow \mathfrak{g}_r^{u_i}$, as well as $a'_i \leftarrow g^{r'_i}$, $\sigma'_{i,1} \leftarrow a_i'^{t'_i} g^{u_i}$, $\mathfrak{f}'_i \leftarrow \mathfrak{g}_r'{}^{u_i}$.
Then, one additionally chooses $x_i, y_i \xleftarrow{\$} \mathbb{Z}_p$ and sets

$$\begin{aligned} \mathfrak{g}_i &\leftarrow \mathfrak{g}_r^{x_i} \quad \mathfrak{h}_i \leftarrow \mathfrak{g}_r^{y_i} & \mathfrak{g}'_i &\leftarrow \mathfrak{g}_r'{}^{x_i} \quad \mathfrak{h}'_i \leftarrow \mathfrak{g}_r'{}^{y_i} \\ \mathfrak{l}_i &\leftarrow (\mathfrak{g}_r^{t_i}/(\mathfrak{g}_i\mathfrak{h}_i^d))^{1/e} & \mathfrak{l}'_i &\leftarrow (\mathfrak{g}_r'{}^{t'_i}/(\mathfrak{g}'_i\mathfrak{h}_i'^d))^{1/e} \\ \overline{C}_i &\leftarrow (g, a_i^e, a_i, a_i^d) & \overline{C}'_i &\leftarrow (g, a_i'^e, a'_i, a_i'^d) \end{aligned}$$

By construction

$$\begin{aligned} \mathfrak{g}_r^{t_i} &= \mathfrak{g}_i\mathfrak{h}_i^d\mathfrak{l}_i^e & \mathfrak{g}_r'{}^{t'_i} &= \mathfrak{g}'_i\mathfrak{h}_i'^d\mathfrak{l}_i'^e \\ \sigma_{i,1} &= a_i^{t_i} g^{u_i} = g^{t_i r_i} \times g^{u_i} & \sigma'_{i,1} &= a_i'^{t'_i} g^{u_i} = g^{t'_i r'_i} \times g^{u_i} \end{aligned}$$

With $\sigma_{i,2} \leftarrow g^{t_i}$ and $\sigma'_{i,2} \leftarrow g^{t'_i}$, σ_i and σ'_i are valid signatures of $(\overline{C}_i, \overline{C}_0)$ with respect to $\mathfrak{g}_r$ and $(\overline{C}'_i, \overline{C}_0)$ with respect to $\mathfrak{g}'_r$, respectively. Eventually, the verification keys $\mathsf{vk}_i = (\mathfrak{f}_i, \mathfrak{l}_i, \mathfrak{g}_i, \mathfrak{h}_i)$ and $\mathsf{vk}'_i = (\mathfrak{f}'_i, \mathfrak{l}'_i, \mathfrak{g}'_i, \mathfrak{h}'_i)$ are correctly related for the secret keys (u_i, v_i, x_i, y_i). From $\mathfrak{l}_i = (\mathfrak{g}_r^{t_i}/(\mathfrak{g}_i\mathfrak{h}_i^d))^{1/e} = \mathfrak{g}_r^{(t_i - x_i - dy_i)/e}$: we have $v_i = (t_i - x_i - dy_i)/e$. From $\mathfrak{l}'_i = (\mathfrak{g}_r'{}^{t'_i}/(\mathfrak{g}'_i\mathfrak{h}_i'^d))^{1/e} = \mathfrak{g}_r'{}^{(t'_i - x_i - dy_i)/e}$: we have $v'_i = (t'_i - x_i - dy_i)/e = (t'_i - t_i)/e + v_i$, which means that $\delta_i = (t'_i - t_i)/e$.
Using the signing key SK, we can complete and sign $\overline{\mathsf{vk}}_i$ (with random R_i) and $\overline{\mathsf{vk}}'_i$ (with random R'_i, which implicitly defines μ_i). As explained above, this perfectly simulates the view of the adversary in the previous game.

Game $\mathbf{G}_2$: Let us be given $\mathsf{Cred}(u_i, g; \mathfrak{g}_r, r_i, t_i)$ and $\mathsf{Cred}(u_i, g; \mathfrak{g}'_r, r'_i, t'_i)$, for random $u_i \xleftarrow{\$} \mathbb{Z}_p$, which provide all the required inputs from the first part of the simulation in the previous game (before choosing v_i, w_i). They all follow the distribution $\mathcal{D}_{g,\mathfrak{g}_r}(u_i, u_i)$. We can thus continue the simulation as above, in a perfectly indistinguishable way.
More precisely, with the above notations, we use as input two tuples

$$\left(\begin{pmatrix} g \\ \mathfrak{g}_r \end{pmatrix}, \begin{pmatrix} g \\ \mathfrak{g}_r \end{pmatrix}^{t_i}, \begin{pmatrix} g \\ g^{t_i} \end{pmatrix}^{r_i} \times \begin{pmatrix} 1 \\ g^{u_i} \end{pmatrix}, \mathfrak{g}_r^{u_i}\right)$$

and

$$\left(\begin{pmatrix} g \\ \mathfrak{g}'_r \end{pmatrix}, \begin{pmatrix} g \\ \mathfrak{g}'_r \end{pmatrix}^{t'_i}, \begin{pmatrix} g \\ g^{t'_i} \end{pmatrix}^{r'_i} \times \begin{pmatrix} 1 \\ g^{u_i} \end{pmatrix}, \mathfrak{g}_r'{}^{u_i}\right)$$

which are $\mathsf{Cred}(u_i, g; \mathfrak{g}_r, r_i, t_i)$ and $\mathsf{Cred}(u_i, g; \mathfrak{g}'_r, r'_i, t'_i)$.

Game $\mathbf{G}_3$: Let us be given two credentials of u_i and u'_i, $\mathsf{Cred}(u_i, g; \mathfrak{g}_r, r_i, t_i)$ and $\mathsf{Cred}(u'_i, g; \mathfrak{g}'_r, r'_i, t'_i)$, for random $u_i, u'_i \xleftarrow{\$} \mathbb{Z}_p$. Inputs follow the distribution $\mathcal{D}_{g,\mathfrak{g}_r}(u_i, u'_i)$. And we do as above. Under the Unlinkability Assumption (see Definition 6) the view is computationally indistinguishable.

Game $\mathbf{G}_4$: We receive a Multi Diffie-Hellman tuple $(\mathfrak{g}_r, \mathfrak{g}_i, \mathfrak{h}_i, \mathfrak{g}'_r, \mathfrak{g}'_i, \mathfrak{h}'_i) \xleftarrow{\$} \mathcal{D}^6_{\mathsf{mdh}}(\mathfrak{g}_r)$. So we know all the scalars, except x_i, y_i and α, which are implicitly defined by the input challenge. Then, by choosing $t_i, t'_i \xleftarrow{\$} \mathbb{Z}_p$, we can define $\mathfrak{l}_i, \mathfrak{l}'_i$ as in the previous game, and the ciphertexts and signatures are generated honestly with random scalars $r_i, r'_i \xleftarrow{\$} \mathbb{Z}_p$.

Game $\mathbf{G}_5$: We now receive $(\mathfrak{g}_r, \mathfrak{g}_i, \mathfrak{h}_i, \mathfrak{g}'_r, \mathfrak{g}'_i, \mathfrak{h}'_i) \xleftarrow{\$} \mathcal{D}^6_{\$}(\mathfrak{g}_r)$. We do the simulation as above. The view of the adversary is indistinguishable under the DDH assumption in $\mathbb{G}_2$.
In this game, $\mathsf{vk}'_i = (\mathfrak{f}_i = {\mathfrak{g}'_r}^{u'_i}, \mathfrak{l}_i = {\mathfrak{g}'_r}^{v'_i}, \mathfrak{g}_i = {\mathfrak{g}'_r}^{x'_i}, \mathfrak{h}_i = {\mathfrak{g}'_r}^{y'_i})$, with $x'_i, y'_i \xleftarrow{\$} \mathbb{Z}_p$ because of the random tuple, $v'_i = v_i + (t'_i - t_i)/e$, for random t'_i and t_i, it is thus also random, and u'_i is chosen at random.

Game $\mathbf{G}_6$: We can now choose at random the signing keys $\mathsf{sk}_i = (u_i, v_i, x_i, y_i)$ and $\mathsf{sk}'_i = (u'_i, v'_i, x'_i, y'_i)$ in order to sign the ciphertexts. □

In this last game, the i-th honest voter is simulated with input/output ciphertexts that are random encryptions of 1, and input/output signing keys (and thus verification keys vk_i and vk'_i) independently random. This is similar to game $\mathbf{G}_0$ for $i+1$. Hence, iterating from $i=1$ to $i=n'$, in the end, all the honest input/output ballots are independently random.

6 Application to Electronic Voting

The use of mix-net for electronic voting implies some particularities that we will discuss in this section: one can optimize the storage of the mix-net in this scenario and one may want to add one more property for the security called *receipt-freeness*.

6.1 Efficiency in case of Electronic Voting

Suppose the ballots are sent successively to a mix-server. The mix-server can compute the round parameters and then, on the fly, perform individual verifications and randomization of each ballot, as well as the product of the $\mathfrak{f}_i$'s adaptively. Eventually, when at the closing time, one just has to do/adapt the global proof of Diffie-Hellman tuple, and then output the ballots in a permuted order.

After the multiple mixing steps, one can remove $\sigma_{i,2}$ and $\Sigma_{i,2}$ in the final ballot-box, as no more mixing will be required. But there is no need to keep the input ballot-box either, excepted the certified verification keys, as the $\mathfrak{f}_i$'s are used in the Diffie-Hellman tuple proof. Anyway, in electronic voting, the verification keys are usually sealed with the voting list. In our construction we can additionally seal the product $\mathfrak{P}$ of the $\mathfrak{f}_i$'s.

We thus recap the elements to keep: system parameters $(6\times\mathbb{G}_1)$, input ballot-box parameters $(1\times\mathbb{G}_2)$, input votes $(n\times(1\times\mathbb{G}_1+6\times\mathbb{G}_2))$, output ballot-box parameters $(1\times\mathbb{G}_2)$, output votes $(n\times(5\times\mathbb{G}_1+6\times\mathbb{G}_2))$, and the Diffie-Hellman proof $(2\times\mathbb{G}_1+2\times\mathbb{G}_2)$. The global storage is thus limited to $8+6n$ elements from $\mathbb{G}_1$ and $4+12n$ elements from $\mathbb{G}_2$, for all the certified input keys and signed output ciphertexts.

We recall that the verification time is just linear in the number of ballots and does not depend on the number of mixing steps.

6.2 Security Guarantees

As explained, soundness and privacy are guaranteed for all the honest users: honest users are sure that their votes are randomized in the output ballot-box, and their input-output ballots are unlinkable. This is of course the most important requirements.

However, since the u_i's are used to guarantee that no ballots are deleted or inserted, this is important those values to be unknown to the mix-server. In case of collusion of some legitimate voters with the mix-server, it is possible to replace the votes of those dishonest voters. In addition, knowing their u_i's the voters can trace their ballots which allows to break the receipt-freeness. In the next section, we explain how to generate the signing keys in a distributed way, so that nobody knows them. This then addresses both above issues: permutation of all the input ciphertexts (for honest and corrupted voters) is guaranteed and receipt-freeness is provided.

In the Appendix B, we propose a second construction that uses the Square Diffie-Hellman tuples $(\mathfrak{g}_r, \mathfrak{A}_i = \mathfrak{g}_r^{w_i}, \mathfrak{B}_i = \mathfrak{A}_i^{w_i})$ to provide non-miscibility. Then, one can use $\prod \mathfrak{A}'_j = (\prod \mathfrak{A}_i)^\alpha$ instead of $\prod \mathfrak{f}'_j$ and $(\prod \mathfrak{f}_i)^\alpha$, in the Diffie-Hellman tuple, to guarantee the permutation of the verification keys. With this modification, the signing keys can be known to the voters, only the w_i's have to be unknown to them, nor to the mix-server. In the Appendix B.5, we explain how to generate them.

6.3 Receipt-Freeness

In electronic voting, receipt-freeness is a hard to achieve security property: the voter should not be able to convince someone of the content of his vote. Since we already said that the initial ballot does not need to be published (removed from the final output), but just the signed vk_i for each voter, the voter cannot exploit his encryption random coins. But he can use $\mathsf{sk}_i = (u_i, v_i, x_i, y_i)$ to prove which randomized ballot corresponds to his vote: indeed, excepted v'_i that gets unknown to him, $u'_i = u_i$, $x'_i = x_i$, and $y'_i = y_i$ in the randomized ballots. After decryption, this will prove his vote. The voter should not know (u_i, v_i, x_i, y_i). This can be easily done by the voter choosing $(u_{i,1}, v_{i,1}, x_{i,1}, y_{i,1}) \xleftarrow{\$} \mathbb{Z}_p^4$ to compute and prove his knowledge of the exponents,

$$\mathfrak{f}_{i,1} = \mathfrak{g}_r^{u_{i,1}}, \mathfrak{l}_{i,1} = \mathfrak{g}_r^{v_{i,1}}, \mathfrak{g}_{i,1} = \mathfrak{g}_r^{x_{i,1}}, \mathfrak{h}_{i,1} = \mathfrak{g}_r^{y_{i,1}}$$

while the signer (authority) chooses $u_{i,2}, v_{i,2}, x_{i,2}, y_{i,2} \xleftarrow{\$} \mathbb{Z}_p^4$ to compute

$$\mathfrak{f}_{i,2} = \mathfrak{g}_r^{u_{i,2}}, \mathfrak{l}_{i,2} = \mathfrak{g}_r^{v_{i,2}}, \mathfrak{g}_{i,2} = \mathfrak{g}_r^{x_{i,2}}, \mathfrak{h}_{i,2} = \mathfrak{g}_r^{y_{i,2}}$$

The latter can also compute

$$\mathfrak{f}_i = \mathfrak{f}_{i,1} \cdot \mathfrak{f}_{i,2}, \mathfrak{l}_i = \mathfrak{l}_{i,1} \cdot \mathfrak{l}_{i,2}, \mathfrak{g}_i = \mathfrak{g}_{i,1} \cdot \mathfrak{g}_{i,2}, \mathfrak{h}_i = \mathfrak{h}_{i,1} \cdot \mathfrak{h}_{i,2}$$

and generate the signature Σ_i, together with zero-knowledge proofs of knowledge of the scalars $u_{i,2}, v_{i,2}, x_{i,2}, y_{i,2}$. Since now the signing key sk_i is split between the voter and the signer, the voter needs to interact with the signer to build σ_i. However, the privacy of the vote is still guaranteed as it is encrypted under EK.

Acknowledgments

This work was supported in part by the European Community's Seventh Framework Programme (FP7/2007-2013 Grant Agreement no. 339563 – CryptoCloud).

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.

BBG05. Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456. Springer, Heidelberg, May 2005.

BBS04. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, Heidelberg, August 2004.

BCCT12. Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In Shafi Goldwasser, editor, *ITCS 2012*, pages 326–349. ACM, January 2012.

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.

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.

BFPV11. Olivier Blazy, Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Signatures on randomizable ciphertexts. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 403–422. Springer, Heidelberg, March 2011.

Boy08. Xavier Boyen. The uber-assumption family (invited talk). In Steven D. Galbraith and Kenneth G. Paterson, editors, *PAIRING 2008*, volume 5209 of *LNCS*, pages 39–56. Springer, Heidelberg, September 2008.

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

Dam92. Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 445–456. Springer, Heidelberg, August 1992.

EHK+13. Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Heidelberg, August 2013.

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.

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

FS01. Jun Furukawa and Kazue Sako. An efficient scheme for proving a shuffle. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 368–387. Springer, Heidelberg, August 2001.

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.

GI08. Jens Groth and Yuval Ishai. Sub-linear zero-knowledge argument for correctness of a shuffle. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 379–396. Springer, Heidelberg, April 2008.

GL07. Jens Groth and Steve Lu. A non-interactive shuffle with pairing based verifiability. In Kaoru Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 51–67. Springer, Heidelberg, December 2007.

Gro10. Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340. Springer, Heidelberg, December 2010.

GS08. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Heidelberg, April 2008.

GW11. Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011.

HT98. Satoshi Hada and Toshiaki Tanaka. On the existence of 3-round zero-knowledge protocols. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 408–423. Springer, Heidelberg, August 1998.

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.

Nef01. C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In Michael K. Reiter and Pierangela Samarati, editors, *ACM CCS 2001*, pages 116–125. ACM Press, November 2001.
Sho97. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997.

A Non-Miscibility with Square Diffie-Hellman Tuples

In Section 3, we proved unforgeability in the generic group model. But actually, as shown in the proof, the simulator can find the explicit linear combination. Hence, the following computational assumption holds in the generic bilinear group model:

Definition 18 (Extractability Assumption). The extractability assumption states that given n valid pairs $(\boldsymbol{M}_j = (M_{j,i})_i, \sigma_j)_j$ under a verification key vk, for any adversary that produces a new valid pair $(\boldsymbol{M} = (M_i)_i, \sigma)$ under vk, there exists an extractor that outputs $(\alpha_j)_j$ such that $\boldsymbol{M} = \prod_j \boldsymbol{M}_j^{\alpha_j}$ and $\sigma = \prod_j \sigma_j^{\alpha_j}$.

Combined to Theorems 10 and 11, this assumption proves non-miscibility of signed-vectors, when Square Diffie-Hellman triples are concatenated.

A.1 Proof of Theorem 10

We stress that in this theorem, the x_i's are random and public (assumed distinct), but the bases g_i's are truly randomly and independently generated. Up to a guess, which is correct with probability greater than $1/n^2$, we can assume that $\alpha_1, \alpha_2 \neq 0$. We are given a discrete logarithm challenge Z, in basis g. We will embed it in either g_1 or g_2, by randomly choosing a bit b:

- if $b = 0$: set $X = Z$, and randomly choose $v \xleftarrow{\$} \mathbb{Z}_p$ and set $Y = g^v$
- if $b = 1$: set $Y = Z$, and randomly choose $u \xleftarrow{\$} \mathbb{Z}_p$ and set $X = g^u$

We set $g_1 \leftarrow X(= g^u)$, $g_2 \leftarrow Y(= g^v)$, with either u or v unknown, and randomly choose $\beta_i \in \mathbb{Z}_p$, for $i = 3, \ldots, n$ to set $g_i \leftarrow g^{\beta_i}$. Eventually, we randomly choose x_i, for $i = 1, \ldots, n$ and output $(g_i, a_i = g_i^{x_i}, b_i = a_i^{x_i})$ together with x_i, to the adversary which outputs $(\alpha_i)_{i=1,\ldots,n}$ such that $(G = \prod g_i^{\alpha_i}, A = \prod a_i^{\alpha_i} = G^x, B = \prod b_i^{\alpha_i} = A^x)$ for some unknown x. We thus have the following relations:

$$\left(\alpha_1 u + \alpha_2 v + \sum_{i=3}^{n} \alpha_i \beta_i\right) \cdot x = \alpha_1 u x_1 + \alpha_2 v x_2 + \sum_{i=3}^{n} \alpha_i \beta_i x_i$$
$$\left(\alpha_1 u x_1 + \alpha_2 v x_2 + \sum_{i=3}^{n} \alpha_i \beta_i x_i\right) \cdot x = \alpha_1 u x_1^2 + \alpha_2 v x_2^2 + \sum_{i=3}^{n} \alpha_i \beta_i x_i^2$$

If we denote $T = \sum_{i=3}^{n} \alpha_i \beta_i$, $U = \sum_{i=3}^{n} \alpha_i \beta_i x_i$, and $V = \sum_{i=3}^{n} \alpha_i \beta_i x_i^2$, that can be computed, we deduce that:

$$(\alpha_1 u x_1 + \alpha_2 v x_2 + U)^2 = (\alpha_1 u + \alpha_2 v + T)(\alpha_1 u x_1^2 + \alpha_2 v x_2^2 + V)$$

which leads to

$$\alpha_1 \alpha_2 (x_1^2 - x_2^2) uv + \alpha_1 (V - 2Ux_1 + Tx_1^2) u + \alpha_2 (V - 2Ux_2 + Tx_2^2) v + (TV - U^2) = 0$$

We consider two cases:

1. $K = \alpha_2 (x_1^2 - x_2^2) v + V - 2Ux_1 + Tx_1^2 = 0 \bmod p$;
2. $K = \alpha_2 (x_1^2 - x_2^2) v + V - 2Ux_1 + Tx_1^2 \neq 0 \bmod p$;

which can be determined by checking whether the equality below holds or not:

$$g^{-(V-2Ux_1+Tx_1^2)/(\alpha_2(x_1^2-x_2^2))} = Y.$$

One can note that case (1) and case (2) are independent of the bit b.

- If the case (1) happens, but $b = 0$, one aborts. If $b = 1$ (which holds with probability $1/2$ independently of the case) then we can compute $v = -(V-2Ux_1+Tx_1^2)/(\alpha_2(x_1^2-x_2^2)) \bmod p$ which is the discrete logarithm of Z in the basis g.
- Otherwise, the case (2) appears. If $b = 1$ one aborts. If $b = 0$ (which holds with probability $1/2$ independently of the case), v is known and we have $\alpha_1 Ku + \alpha_2(V - 2Ux_2 + Tx_2^2)v + (TV - U^2) = 0 \bmod p$, which means that the discrete logarithm of Z in the basis g is $u = -(\alpha_2(V - 2Ux_2 + Tx_2^2)v + (TV - U^2))/(\alpha_1 K) \bmod p$. □

A.2 Proof of Theorem 11

Lemma 19. *Given any fixed value $\alpha \in \mathbb{Z}_p$ and n valid Square Diffie-Hellman tuples $(g, a_i = g^{x_i}, b_i = a_i^{x_i})$, for any $g \in \mathbb{G}$ and random $x_i \in \mathbb{Z}_p$, outputting $(\alpha_i)_{i=1,\ldots,n}$ such that $\alpha = \sum_{i=1}^n \alpha_i x_i$, with at least one non-zero coefficient α_i, is computationally hard under the* SDL *assumption.*

Proof. Up to a guess, which is correct with probability greater than $1/n$, we can assume that $\alpha_1 \neq 0$. We are given a square discrete logarithm challenge $(g, Z_1 = g^z, Z_2 = g^{z^2})$, in basis g. We set $a_1 \leftarrow Z_1$, $b_1 \leftarrow Z_2$, and randomly choose $x_i \xleftarrow{\$} \mathbb{Z}_p$, for $i = 2, \ldots, n$ to set $(a_i \leftarrow g^{x_i}, b_i \leftarrow a_i^{x_i})$. We then output (g, a_i, b_i), $i = 1, \ldots, n$, to the adversary which outputs $(\alpha_i)_{i=1,\ldots,n}$ and α such that $\alpha_1 z + \sum_{i=2}^n \alpha_i x_i = \alpha$. At this stage, we solve the square discrete logarithm problem by returning $z = (\alpha - \sum_{i=2}^n \alpha_i x_i)/\alpha_1 \bmod p$. □

We now come back to the proof of the theorem. Again, up to a guess, which is correct with probability greater than $1/n$, we can assume that $\alpha_1 \neq 0$. We are given a square discrete logarithm challenge $(g, Z_1 = g^z, Z_2 = g^{z^2})$, in basis g. We set $a_1 \leftarrow Z_1$, $a_2 \leftarrow Z_2$, and randomly choose $x_i \xleftarrow{\$} \mathbb{Z}_p$, for $i = 2, \ldots, n$ to set $(a_i \leftarrow g^{x_i}, b_i = a_i^{x_i})$. We then output (g, a_i, b_i), $i = 2, \ldots, n$, to the adversary that outputs $(\alpha_i)_{i=1,\ldots,n}$ such that $(G = \prod g^{\alpha_i}, A = \prod a_i^{\alpha_i} = G^x, B = \prod b_i^{\alpha_i} = A^x)$ for some unknown x. We thus have the following relations:

$$\left(\sum_{i=1}^n \alpha_i\right) \cdot x = \alpha_1 z + \sum_{i=2}^n \alpha_i x_i \qquad \left(\sum_{i=1}^n \alpha_i\right) \cdot x^2 = \alpha_1 z^2 + \sum_{i=2}^n \alpha_i x_i^2$$

which leads to

$$\left(\alpha_1 z + \sum_{i=2}^n \alpha_i x_i\right)^2 = \left(\alpha_1 + \sum_{i=2}^n \alpha_i\right) \times \left(\alpha_1 z^2 + \sum_{i=2}^n \alpha_i x_i^2\right).$$

If we denote $T = \sum_{i=2}^n \alpha_i x_i$, $U = \sum_{i=2}^n \alpha_i$, and $V = \sum_{i=2}^n \alpha_i x_i^2$, that can be computed from above scalars, we have $(\alpha_1 z + T)^2 = (\alpha_1 + U) \cdot (\alpha_1 z^2 + V)$, and thus

$$U\alpha_1 z^2 - 2T\alpha_1 z + (\alpha_1 + U)V - T^2 = 0 \bmod p.$$

Using Lemma 19 on the $n - 1$ tuples (g, a_i, b_i), for $i = 2, \ldots, n$, the probability that $T = \sum_{i=2}^n \alpha_i x_i = 0$ is negligible, unless one can break the SDL Assumption. So we have $T \neq 0$, with two cases:

1. If $U \neq 0$ then, because computing square roots in $\mathbb{Z}_p$ is easy, one can solve the above quadratic equation for z that admits solutions, and obtain two solutions for z. By testing which one satisfies $g^z = Z_1$, one can find out the correct z and thus solve the SDL problem.
2. If $U = 0$, one can compute $z = (\alpha_1 V - T^2)/(2T\alpha_1) \bmod p$ and thus solve the SDL problem.

□

B Mix-Networks with Squares Diffie-Hellman Tuples

Instead of using the signature scheme from Section 3.5, with the random R_λ to avoid miscibility, one can concatenate Square Diffie-Hellman tuples $(\mathfrak{g}_r, \mathfrak{A}_i = \mathfrak{g}_r^{w_i}, \mathfrak{B}_i = \mathfrak{A}_i^{w_i})$ with a proof of correctness, using Groth-Sahai technique: $\mathsf{Com}_i = (c_i = v_{2,1}^{w_i} v_{1,1}^{\mu_i}, d_i = v_{2,2}^{w_i} v_{1,2}^{\mu_i} g^{w_i}) \in \mathbb{G}_1^2$, a commitment of w_i, then $(\Theta_i = \mathfrak{g}_r^{\mu_i}, \Psi_i = \mathfrak{A}_i^{\mu_i}) \in \mathbb{G}_2^2$ satisfies:

$$e(c_i, \mathfrak{g}_r) = e(v_{2,1}, \mathfrak{A}_i) \cdot e(v_{1,1}, \Theta_i) \qquad e(d_i, \mathfrak{g}_r) = e(v_{2,2} g, \mathfrak{A}_i) \cdot e(v_{1,2}, \Theta_i)$$
$$e(c_i, \mathfrak{A}_i) = e(v_{2,1}, \mathfrak{B}_i) \cdot e(v_{1,1}, \Psi_i) \qquad e(d_i, \mathfrak{A}_i) = e(v_{2,2} g, \mathfrak{B}_i) \cdot e(v_{1,2}, \Psi_i)$$

In this case, one can use $\prod \mathfrak{A}'_j = (\prod \mathfrak{A}_i)^\alpha$ instead of $\prod \mathfrak{f}'_j$ and $(\prod \mathfrak{f}_i)^\alpha$, in the Diffie-Hellman tuple, to guarantee the permutation of the verification keys. Indeed, the voter does not need to know w_i to sign his vote, as explained later.

This leads to the protocol presented on Figure 5. We first give an intuition of the construction, then provide a new proof for permutation of the verification keys.

Parameters

For a security parameter κ, let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, \mathfrak{g}, e) \leftarrow \mathcal{G}(\kappa)$, where g and $\mathfrak{g}$ are random generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively, and $(v_{1,1}, v_{1,2}, v_{2,1}, v_{2,2})$ a Diffie-Hellman tuple, then the global parameters are $\mathsf{Gparam} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, \mathfrak{g}, e, v_{1,1}, v_{1,2}, v_{2,1}, v_{2,2})$;

Let the system keys be $\mathsf{SK} = (S_1, S_2, S_3, S_4, S_5, S_6, S_7) \xleftarrow{\$} \mathbb{Z}_p^7$ the signing key, $\mathsf{VK} = (g_i = g^{S_i})_i$ the associated verification key, $\mathsf{EK} = h = g^d$ the encryption key associated to the decryption key $\mathsf{DK} = d \xleftarrow{\$} \mathbb{Z}_p$, and $\ell = g^e$, for $e \xleftarrow{\$} \mathbb{Z}_p$;

The round parameters are initialized as $\mathsf{Rparam} = (\mathfrak{g}_r = \mathfrak{g}, \mathfrak{s}_r = \mathfrak{s} = \mathfrak{g}^{S_3})$.

Initialization $(r = 0)$

- Keys: $\mathsf{sk}_i = (u_i, v_i, x_i, y_i; w_i) \xleftarrow{\$} \mathbb{Z}_p^5$ and $\mathsf{vk}_i = (\mathfrak{g}_r; \mathfrak{f}_i = \mathfrak{g}_r^{u_i}, \mathfrak{l}_i = \mathfrak{g}_r^{v_i}, \mathfrak{g}_i = \mathfrak{g}_r^{x_i}, \mathfrak{h}_i = \mathfrak{g}_r^{y_i}; \mathfrak{A}_i = \mathfrak{g}_r^{w_i}, \mathfrak{B}_i = \mathfrak{A}_i^{w_i})$, with a system signature $\Sigma_i = \mathfrak{g}_r^{S_1} \cdot \mathfrak{f}_i^{S_2} \mathfrak{l}_i^{S_3} \mathfrak{g}_i^{S_4} \mathfrak{h}_i^{S_5} \cdot \mathfrak{A}_i^{S_6} \mathfrak{B}_i^{S_7}$ and a proof $\pi_i = (\mathsf{Com}_i, \mathsf{Proof}_i)$ of Square Diffie-Hellman for the tuple $(\mathfrak{g}_r, \mathfrak{A}_i, \mathfrak{B}_i)$: $\mathsf{Com}_i = (c_i = v_{2,1}^{w_i} v_{1,1}^{\mu_i}, d_i = v_{2,2}^{w_i} v_{1,2}^{\mu_i} g^{w_i}) \in \mathbb{G}_1^2$, a commitment of w_i and $\mathsf{Proof}_i = (\Theta_i = \mathfrak{g}_r^{\mu_i}, \Psi_i = \mathfrak{A}_i^{\mu_i}) \in \mathbb{G}_2^2$;
- Ciphertexts: $C_i = (g, \ell_i = \ell^{r_i}, a_i = g^{r_i}, b_i = h^{r_i} M_i)$, $C_0 = (1, \ell, g, h)$, with two signatures $\sigma_i = g^{u_i} \ell_i^{v_i} a_i^{x_i} b_i^{y_i}$ of C_i and $\tau_i = \ell^{v_i} g^{x_i} h^{y_i}$ of C_0.

In addition to the parameters $(\mathsf{Gparam}, \mathsf{VK}, \mathsf{Rparam})$, the authenticated ciphertext consists of 7 elements from $\mathbb{G}_1$ and 9 elements from $\mathbb{G}_2$:

$$(a_i, b_i, \ell_i, \sigma_i, \tau_i) \in \mathbb{G}_1^5, (\mathfrak{f}_i, \mathfrak{g}_i, \mathfrak{h}_i, \mathfrak{l}_i; \mathfrak{A}_i, \mathfrak{B}_i) \in \mathbb{G}_2^6, \Sigma_i \in \mathbb{G}_2, (c_i, d_i, \Theta_i, \Psi_i) \in \mathbb{G}_1^2 \times \mathbb{G}_2^2.$$

Mixing

The mix-server chooses random scalars $\alpha \xleftarrow{\$} \mathbb{Z}_p$, and publishes new round parameters $\mathsf{Rparam}' = (\mathfrak{g}'_r = \mathfrak{g}_r^\alpha, \mathfrak{s}'_r = \mathfrak{s}_r^\alpha)$: for each ballot, it chooses $\gamma_i, \delta_i, z_i \xleftarrow{\$} \mathbb{Z}_p$, and sets

$$\mathsf{vk}'_i = (\mathfrak{g}'_r; \mathfrak{f}'_i = \mathfrak{f}_i^\alpha, \mathfrak{l}'_i = \mathfrak{l}_i^\alpha \mathfrak{g}_r'^{\delta_i}, \mathfrak{g}'_i = \mathfrak{g}_i^\alpha, \mathfrak{h}'_i = \mathfrak{h}_i^\alpha; \mathfrak{A}'_i = \mathfrak{A}_i^\alpha, \mathfrak{B}'_i = \mathfrak{B}_i^\alpha)$$
$$C'_i = (g, \ell'_i = \ell_i \cdot \ell^{\gamma_i}, a'_i = a_i \cdot g^{\gamma_i}, b'_i = b_i \cdot h^{\gamma_i})$$
$$\Sigma'_i = \Sigma_i^\alpha \cdot \mathfrak{s}_r'^{\delta_i} \qquad \sigma'_i = \sigma_i \cdot \tau_i^{\gamma_i} \cdot \ell_i'^{\delta_i} \qquad \tau'_i = \tau_i \cdot \ell_r'^{\delta_i}$$
$$c'_i = c_i \cdot v_{1,1}^{z_i} \qquad d'_i = d_i \cdot v_{1,2}^{z_i} \qquad \Theta'_i = (\Theta_i \cdot \mathfrak{g}_r^{z_i})^\alpha \qquad \Psi'_i = (\Psi_i \cdot \mathfrak{A}_i^{z_i})^\alpha$$

Output all the output ballots in random order with the round parameters and the global proof π' of Diffie-Hellman tuple for $(\mathfrak{g}_r, \mathfrak{g}'_r, \prod_i \mathfrak{A}_i, \prod_i \mathfrak{A}'_i)$.

Verification

The verifier will check this proof π, the number of input tuples is the same as the number of output tuples, the $\mathfrak{A}_i$ are all distinct and different of 1, and $\sigma_i, \Sigma_i, \pi_i$ are valid on individual output tuples.

Figure 5. Shuffling of ElGamal Ciphertexts

B.1 Intuition

This construction will take advantage of the signature σ_i of C_i under the verification key vk_i, also signed in Σ_i under the system verification key VK, thanks to the linear homomorphism. For the intuition of soundness, we assume $\delta_i = 0$ (see Figure 5) to have a perfect linear randomization (such linearity can lead to linkability, hence the alteration δ_i which will allow the zero-knowledge property, that will be formally proven later). The mix-server will indeed be able to generate a randomization C'_i of C_i, and a signature σ'_i under the key vk_i. Then, one will note that σ'_i is still a valid signature of C'_i, but under $\mathsf{vk}'_i = \mathsf{vk}_i^\alpha$ (we stress that for this intuition, $\delta_i = 0$) and new parameters, which is also signed by $\Sigma'_i = \Sigma_i^\alpha$, under the same system verification key VK, thanks to the linear homomorphism. Several properties will be required, with the evolving round parameters $\mathsf{Rparam} = (\mathfrak{g}_r, \mathfrak{s}_r)$ initially set to $(\mathfrak{g}, \mathfrak{s})$, where $\mathfrak{s}$ is a signature of $\mathfrak{g}$ under g_3 as $e(g, \mathfrak{s}) = e(g_3, \mathfrak{g})$. Note that this signature $\mathfrak{s}$ will just be useful when $\delta_i \neq 0$, to break linearity properties that could be used by an adversary. We expect the operations performed by the mix-server guarantee:

1. only randomizations of some C_i can be signed under a new verification key vk'_i. To this aim, the sender will sign the ciphertext $(g, \ell^{r_i}, g^{r_i}, h^{r_i} M_i)$ of M_i and the ciphertext $(1, \ell, g, h)$ of 1, with its signing key sk_i (associated to the verification key vk_i). These are ElGamal ciphertexts, with a first component that impose the randomization of the ciphertext and the second component that excludes linear attacks (when $\delta_i \neq 0$). Indeed, since only linear combinations are possible, one can just sign some pair $C'_i = (g^{t_i}, \ell'_i = \ell^{t_i r_i + \gamma_i}, a'_i = g^{t_i r_i + \gamma_i}, b'_i = h^{t_i r_i + \gamma_i} M_i^{t_i})$, which is a randomization of $C_i^{t_i}$. If we impose the first component to remain g, then $t_i = 1$, and C'_i is a randomization of C_i. If one denotes σ'_i the signature of this randomization under vk_i, one notes this is also a signature of C'_i under $\mathsf{vk}'_i = \mathsf{vk}_i^{\alpha_i}$ (when $\delta_i \neq 0$, thanks to $\mathfrak{s}$, one can alter vk'_i and adapt σ'_i), for a new public parameter $\mathfrak{g}'_r = \mathfrak{g}_r^{\alpha_i}$. If $\mathfrak{g}'_r$ is required to be common for all the ciphertexts, then $\alpha_i = \alpha$ is a constant. We can also consider the encryption C_r of 1, and the signature τ_i:

$$e(\sigma'_i, \mathfrak{g}'_r) = e(g, \mathfrak{f}_i^\alpha) \cdot e(\ell'_i, \mathfrak{l}_i^\alpha) \cdot e(a'_i, \mathfrak{g}_i^\alpha) \cdot e(b'_i, \mathfrak{h}_i^\alpha)$$
$$e(\tau_i, \mathfrak{g}'_r) = e(\ell, \mathfrak{l}_i^\alpha) \cdot e(g, \mathfrak{g}_i^\alpha) \cdot e(h, \mathfrak{h}_i^\alpha)$$

2. one needs non-miscibility of the verifications keys to guarantee $\mathsf{vk}'_i = \mathsf{vk}_i^{\alpha_i}$ (when $\delta_i = 0$). To this aim, we will consider $\mathsf{vk}_i = (\mathfrak{g}_r; \mathfrak{f}_i, \mathfrak{l}_i, \mathfrak{g}_i, \mathfrak{h}_i; \mathfrak{g}_r^{w_i}, \mathfrak{g}_r^{w_i^2})$ for signing quadruples (the above ciphertexts C_0 and C_i) with the four components $(\mathfrak{f}_i, \mathfrak{l}_i, \mathfrak{g}_i, \mathfrak{h}_i)$, while the three other components $(\mathfrak{g}_r, \mathfrak{g}_r^{w_i}, \mathfrak{g}_r^{w_i^2})$ guarantee non-miscibility, for unknown and random $w_i \xleftarrow{\$} \mathbb{Z}_p$, as shown in Theorem 11. C_i is converted into C'_i signed under the new verification key $\mathsf{vk}'_i = (\mathfrak{g}_r^\alpha; \mathfrak{f}_i^\alpha, \mathfrak{l}_i^\alpha, \mathfrak{g}_i^\alpha, \mathfrak{h}_i^\alpha; \mathfrak{g}_r^{\alpha w_i}, \mathfrak{g}_r^{\alpha w_i^2})$, with a valid signature $\Sigma'_i = \Sigma_i^\alpha$ (still when $\delta_i = 0$), under VK:

$$e(g, \Sigma'_i) = e(g_1, \mathfrak{g}_r^\alpha) \cdot e(g_2, \mathfrak{f}_i^\alpha) e(g_3, \mathfrak{l}_i^\alpha) e(g_4, \mathfrak{g}_i^\alpha) e(g_5, \mathfrak{h}_i^\alpha) \cdot e(g_6, \mathfrak{A}_i^\alpha) e(g_7, \mathfrak{B}_i^\alpha)$$

For proving the Square Diffie-Hellman tuples, one needs the validity proof, that is initially known for vk_i with $\pi_i = (\mathsf{Com}_i = (c_i, d_i), \mathsf{Proof}_i = (\Theta_i, \Psi_i))$ to be converted into $\pi'_i = (\mathsf{Com}'_i = (c'_i, d'_i), \mathsf{Proof}'_i = (\Theta'_i, \Psi'_i))$. With the computations given in Figure 5, one has

$$e(c'_i, \mathfrak{g}_r^\alpha) = e(v_{2,1}, \mathfrak{A}_i^\alpha) \cdot e(v_{1,1}, \Theta'_i) \qquad e(d_i, \mathfrak{g}_r^\alpha) = e(v_{2,2} g, \mathfrak{A}_i^\alpha) \cdot e(v_{1,2}, \Theta'_i)$$
$$e(c'_i, \mathfrak{A}_i^\alpha) = e(v_{2,1}, \mathfrak{B}_i^\alpha) \cdot e(v_{1,1}, \Psi'_i) \qquad e(d'_i, \mathfrak{A}_i^\alpha) = e(v_{2,2} g, \mathfrak{B}_i^\alpha) \cdot e(v_{1,2}, \Psi'_i)$$

In the initial proof Π_i, Com_i is a Groth-Sahai [GS08] commitment of w_i and Proof_i is the proof that both $\mathfrak{A}_i = \mathfrak{g}_r^{w_i}$ and $\mathfrak{B}_i = \mathfrak{A}_i^{w_i}$. Then, one just randomize the commitment Com_i into Com'_i of the same value w_i, which allows the update of Proof_i into Proof'_i, as detailed in figure 5.

3. a ciphertext C_i can only appear once in an output C'_i. With the above notation, to avoid duplication, since we imposed the constant α, one can just check there are no collisions among the components $\mathfrak{A}'_i$ in the vk'_i (assuming there was no collisions among the components $\mathfrak{A}_i$ in the vk_i's).
4. all the input ciphertexts C_i should appear in an output C'_i. To this aim, one could think this is enough to count them, and have as many output tuples as input tuples, but the server could collude with a legitimate sender (who did not contribute yet in the initial set of ballots, but has all the material) and replace a ballot by a new legitimate ballot for this new user. To avoid that, one also has to check that $\prod_i \mathfrak{A}'_i = (\prod \mathfrak{A}_i)^\alpha$, which guarantees the same w_i's are in both products: indeed, as all individual ballots are signed, they must contain their original w_i. In the example of electronic voting, this exclude the mix-server with voting right (or colluding with legitimate voters) to replace ballots in the initial ballot-box by new ballots.

If one denotes $(\mathfrak{g}'_r = \mathfrak{g}_r^\alpha, \mathfrak{s}'_r = \mathfrak{s}_r^\alpha, \mathfrak{f}'_i = \mathfrak{f}_i^\alpha, \mathfrak{g}'_i = \mathfrak{g}_i^\alpha, \mathfrak{h}'_i = \mathfrak{h}_i^\alpha, \mathfrak{A}'_i = \mathfrak{A}_i^\alpha, \mathfrak{B}'_i = \mathfrak{B}_i^\alpha)$, the new tuples $(a'_i, b'_i, \ell'_i, \sigma'_i, \tau_i)$, $(\mathfrak{g}'_r; \mathfrak{f}'_i, \mathfrak{l}'_i, \mathfrak{g}'_i, \mathfrak{h}'_i; \mathfrak{A}'_i, \mathfrak{B}'_i)$, Σ'_i, $(c'_i, d'_i, \Theta'_i, \Psi'_i)$ satisfy the initial relations (signatures and Square Diffie-Hellman proofs), with respect to the new round parameters $\mathsf{Rparam}' = (\mathfrak{g}'_r, \mathfrak{s}'_r)$. Everything is random, except τ_i: hence the need to randomize it with $\delta_i \neq 0$. Then, one can re-iterate the mixing process.

B.2 Efficiency

As shown in Figure 5, from n ballots, that each consists of 7 group elements from $\mathbb{G}_1$ and 9 group elements from $\mathbb{G}_2$, the mix-server has to perform 8 exponentiations in $\mathbb{G}_1$ and 13 exponentiations in $\mathbb{G}_2$ per ballot to randomize them (most of them to the same exponent α), and the overhead is constant: 2 exponentiation in $\mathbb{G}_2$ for Rparam' and just one zero-knowledge proof of Diffie-Hellman tuple in $\mathbb{G}_2$. One can note that the overall verification just requires individual checks of ballots, which can be performed on individual ballots in any order and even on independent machines, except the total numbers of input and output ciphertexts that have to be the same, the absence of collisions among the $\mathfrak{A}'_i$, as well as the Diffie-Hellman relation proof for $(\mathfrak{g}_r, \mathfrak{g}'_r, \prod \mathfrak{A}_i, \prod \mathfrak{A}'_i)$.

In addition, since mixing should be iterated multiple times to guarantee privacy in case of some malicious mix-servers, one can note that in our mechanism, one just needs to output the initial ballot-box $\mathcal{BB}\mathsf{ox}^{(0)}$ and the final ballot-box $\mathcal{BB}\mathsf{ox}^{(N)}$ (no need of keeping the intermediate ones) and the Diffie-Hellman relation proof for $(\mathfrak{g}_r^{(0)}, \mathfrak{g}_r^{(N)}, \mathfrak{A}^{(0)} = \prod \mathfrak{A}_i, \mathfrak{A}^{(N)} = \prod \mathfrak{A}_i^{(N)})$, which can either be of linear size in the number of iterations if one keeps each individual proof for $(\mathfrak{g}_r, \mathfrak{g}'_r, \prod \mathfrak{A}_i, \prod \mathfrak{A}'_i)$ or just constant size if one uses Groth-Sahai proofs, that can be updated.

Indeed, if we consider the Groth-Sahai proof $\pi_k = (\mathsf{Com}^{(k)}, \Theta^{(k)}, \Psi^{(k)})$ of Diffie-Hellman for $(\mathfrak{g}_r^{(0)}, \mathfrak{g}_r^{(k)}, \mathfrak{A}^{(0)}, \mathfrak{A}^{(k)} = \prod \mathfrak{A}_i^{(k)})$ at round k, $\mathsf{Com}^{(k)}$ is thus a commitment of α_k where α_k satisfies $\mathfrak{g}_r^{(k)} = (\mathfrak{g}_r^{(0)})^{\alpha_k}$: $\mathsf{Com}^{(k)} = \mathsf{Com}(\alpha_k, \mu_k)$. To update the proof for the next round, one can construct (as explained on Figure 6) a commitment $\mathsf{Com}^{(k+1)} = \mathsf{Com}(\alpha_k \times \alpha, \mu_k \alpha_k + \mu)$, where α is the global power in the mixing and $\mu \xleftarrow{\$} \mathbb{Z}_p$ is a randomization of the commitment, and appropriately update $\Theta^{(k)}$ and $\Psi^{(k)}$ into $\Theta^{(k+1)}$ and $\Psi^{(k+1)}$. By this way, we obtain a constant overhead, whatever the number of rounds.

Eventually, one even does not need to publish the entire initial ballot-box, but just the $\mathfrak{A}_i$'s of all the voters: they can be put in real-time on a public-board so that each voter can check that his vote has been added to the ballot-box. After multiple mixings, one just needs to publish the randomized and permuted ballots, with the global proof of Diffie-Hellman tuple for $(\mathfrak{g}_r^{(0)} = \mathfrak{g}, \mathfrak{g}_r^{(N)}, \mathfrak{A}^{(0)} = \prod \mathfrak{A}_i, \mathfrak{A}^{(N)} = \prod \mathfrak{A}_i^{(N)})$.

We summarize all these remarks in Figure 6, where $\mathcal{BB}\mathsf{ox}^{(k)}$ denotes the ballot-box after round k, $\mathfrak{A}^{(0)} = \prod \mathfrak{A}_i$ is the product of all the initial $\mathfrak{A}_i$'s and $\alpha_k = \prod \alpha_j$ the product of all the

From round k to round $k+1$ (with the use of Groth-Sahai proofs)

1. one computes $\mathcal{BB}\mathsf{ox}^{(k+1)}$ from $\mathcal{BB}\mathsf{ox}^{(k)}$ with the Mixing of a shuffle with α;
2. from the Diffie-Hellman proof for $(\mathfrak{g}_r^{(0)}, \mathfrak{g}_r^{(k)}, \mathfrak{A}^{(0)}, \mathfrak{A}^{(k)} = \prod \mathfrak{A}_i^{(k)})$,

$$\mathsf{Com}^{(k)} = (c = v_{2,1}^{\alpha_k} v_{1,1}^{\mu_k}, d = v_{2,2}^{\alpha_k} v_{1,2}^{\mu_k} g^{\alpha_k}) \qquad \Theta^{(k)} = (\mathfrak{g}_r^{(k)})^{\mu_k} \qquad \Psi^{(k)} = (\mathfrak{A}^{(k)})^{\mu_k}$$

one can compute the proof for $(\mathfrak{g}_r^{(0)}, \mathfrak{g}_r^{(k+1)}, \mathfrak{A}^{(0)}, \mathfrak{A}^{(k+1)})$, for random $\mu \xleftarrow{\$} \mathbb{Z}_p$:

$$\mathsf{Com}^{(k+1)} = (c^{\alpha} \cdot v_{1,1}^{\mu}, d^{\alpha} \cdot v_{1,2}^{\mu})$$
$$\Theta^{(k+1)} = (\Theta^{(k)})^{\alpha} \cdot (\mathfrak{g}_r^{(k)})^{\mu} \qquad \Psi^{(k+1)} = (\Psi^{(k)})^{\alpha} \cdot (\mathfrak{A}^{(k+1)})^{\mu}$$

3. one can destroy $\mathcal{BB}\mathsf{ox}^{(k)}$.

Last round N

One can also forget $C_0^{(N)}$ with its signatures $\tau_i^{(N)}$ in $\mathcal{BB}\mathsf{ox}^{(N)}$. Hence, for a mix-net of N rounds, one just have to keep: $\{\mathfrak{A}_i^{(0)}\}, \mathcal{BB}\mathsf{ox}^{(N)}, \mathsf{Com}^{(N)}, \Theta^{(N)}, \Psi^{(N)}$ which consists of $6n+3$ elements in $\mathbb{G}_1$ and $10n+4$ elements in $\mathbb{G}_2$ for n ciphertexts.

Figure 6. Efficient Mix-Net with Multiple Rounds

powers used for mixing since the first round. This mix-net becomes the most-efficient mix-net with verification complexity independent of the number of mixing rounds, to the best of our knowledge.

B.3 Security Analysis

The soundness proof can be improved for the second step, using the extractability assumption, for the permutation of the verification keys. Indeed, note that the mix-server will add the proof that $(\mathfrak{g}_r, \mathfrak{g}'_r, \prod_j \mathfrak{A}_j, \prod_i \mathfrak{A}'_i)$ is a Diffie-Hellman tuple where the products are on the ballots in the initial ballot-box for the $\mathfrak{A}_j$'s and in the output ballot-box for the $\mathfrak{A}'_i$'s. From the extractability assumption, an extractor provides this α such that $\mathfrak{g}'_r = \mathfrak{g}_r^{\alpha}$, then $\prod_i \mathfrak{A}'_i = (\prod_j \mathfrak{A}_j)^{\alpha}$. As proven above, for each i (in the output ballot-box), there exists j_i among the legitimate keys such that $\mathfrak{A}'_i = \mathfrak{A}_{j_i}^{\alpha}$. Hence, we have $(\prod_i \mathfrak{A}_{j_i})^{\alpha} = \prod_i \mathfrak{A}'_i = (\prod_j \mathfrak{A}_j)^{\alpha}$. As a consequence, $\prod_i \mathfrak{A}_{j_i} = \prod_j \mathfrak{A}_j$. Let us now assume that with non-negligible probability, a malicious mix-server manages to insert an w_{j_i} that was not in the initial ballot-box.

We will again exploit the fact that nobody knows the w_i's in the vk_i's, while they are guaranteed to be randomly drawn (we will show later how to make it):

- First, we modify the generation of the proofs of Square Diffie-Hellman tuples. To this aim, we change the global parameters, with a non-Diffie-Hellman tuple $(v_{1,1}, v_{1,2}, v_{2,1}, v_{2,2})$, so that $(v_{1,1}, v_{1,2}, v_{2,1}, g \cdot v_{2,2})$ is a Diffie-Hellman tuple, making the Groth-Sahai commitments perfectly hiding. Under the DDH assumption in $\mathbb{G}_1$, this makes no difference.
- This is equivalent to explicitly set $(v_{1,1}, v_{1,2}, v_{2,1} = v_{1,1}^{\rho}, v_{2,2} = v_{1,2}^{\rho}/g)$ with a random known scalar ρ, and then, so that $(v_{1,1}, v_{1,2}, v_{2,1}, g \cdot v_{2,2})$ is a Diffie-Hellman tuple, making the Groth-Sahai commitments perfectly hiding. We can indeed see the commitment of w_i with randomness μ_i as $(c_i, d_i) = (v_{1,1}^{\mu_i + \rho w_i}, v_{1,2}^{\mu_i + \rho w_i})$, the commitment of 0 with randomness $\nu_i = \mu_i + \rho w_i$. Hence, $\Theta_i = \mathfrak{g}_r^{\mu_i} = \mathfrak{g}_r^{\nu_i - \rho w_i} = \mathfrak{A}_i^{-\rho} \cdot \mathfrak{g}_r^{\nu_i}$ and similarly $\Psi_i = \mathfrak{A}_r^{\mu_i} = \mathfrak{A}_r^{\nu_i - \rho w_i} = \mathfrak{B}_i^{-\rho} \cdot \mathfrak{A}_r^{\nu_i}$.
- This new setup allows to simulate the proofs without knowing w_i, and even for random non Square Diffie-Hellman tuples, which are indistinguishable from Square Diffie-Hellman tuples under the DSDH assumption in $\mathbb{G}_2$.
- Eventually, we choose $\mathfrak{A}_i \leftarrow \mathfrak{g}^{w_i} \mathfrak{h}^{y_i}$, and $\mathfrak{B}_i \leftarrow \mathfrak{g}^{z_i}$ for random scalars $w_i, y_i, z_i \xleftarrow{\$} \mathbb{Z}_p$, with an independent group element $\mathfrak{h} \xleftarrow{\$} \mathbb{G}_2$.

In this last game, a successful modification of the ballots leads to

$$\mathfrak{g}^{\sum_i w_{j_i}}\mathfrak{h}^{\sum_i y_{j_i}} = \prod_i \mathfrak{g}^{w_{j_i}}\mathfrak{h}^{y_{j_i}} = \prod_i \mathfrak{A}_{j_i} = \prod_j \mathfrak{A}_j = \prod_j \mathfrak{g}^{w_j}\mathfrak{h}^{y_j} = \mathfrak{g}^{\sum_j w_j}\mathfrak{h}^{\sum_j y_j}.$$

The left-hand side sum is on $\mathcal{I}$, the set of the indices in the output ballot-box, and the right-hand side sum is on $\mathcal{J}$, the set of the indices in the input ballot-box. Again, because of the collision checks on the $\mathfrak{A}'_i$'s, there is no repetition in the sets, because of the equal sizes of the ballot boxes, the two sets have the same cardinalities. Let us denote $\mathcal{I}' = \mathcal{I}\backslash(\mathcal{I}\cap\mathcal{J})$ and $\mathcal{J}' = \mathcal{J}\backslash(\mathcal{I}\cap\mathcal{J})$. In case of misbehavior of the mix-server, one gets $\mathcal{I}' \neq \emptyset$ and $\mathcal{J}' \neq \emptyset$ but

$$\mathfrak{g}^{\sum_{i\in\mathcal{I}'} w_{j_i}}\mathfrak{h}^{\sum_{i\in\mathcal{I}'} y_{j_i}} = \mathfrak{g}^{\sum_{j\in\mathcal{J}'} w_j}\mathfrak{h}^{\sum_{j\in\mathcal{J}'} y_j}$$

Because of the unpredictability of the y_i's in the $\mathfrak{A}_i$'s, with overwhelming probability $\sum_{i\in\mathcal{I}'} y_{j_i} \neq \sum_{j\in\mathcal{J}'} y_j$, which provides the discrete logarithm of $\mathfrak{h}$ is basis $\mathfrak{g}$.

B.4 Unlinkability

This construction with Square Diffie-Hellman tuples adds group elements of $\mathbb{G}_2$ in vk_i. They provide non-miscibility and thus the soundness but by adding elements, unlinkability could be not verified anymore. This section proves that this is not the case: the unlinkability still remains. The security proof will follow the proof in Section 5: new games are needed at the beginning to randomize the Square Diffie-Hellman tuples and the Groth-Sahai proofs in the input ballot-box, the middle of the proof is exactly the same and finally a last game is needed at the end to randomize the Square Diffie-Hellman tuple in the output ballot-box.

Again we stress that in the following proof, we assume (at least) two input ballots to have been generated by honest (non-compromised) voters. More precisely, at least two signing keys should not be known to the adversary, nor the random coins used by the mix-server. We start from the initial real game that generates the view of the adversary, for the input ballots and the randomized ballots, and conclude with a game where the randomized ballots are all generated randomly, independently of the input ballots: b is thus perfectly hidden.

Game $\mathbf{G}_0$: One generates a group structure $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, \mathfrak{g}, e) \leftarrow \mathcal{G}(\kappa)$, and a Diffie-Hellman tuple $(v_{1,1}, v_{1,2}, v_{2,1}, v_{2,2})$ in $\mathbb{G}_1$. One then sets the global parameters, $\mathsf{Gparam} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, \mathfrak{g}, e, v_{1,1}, v_{1,2}, v_{2,1}, v_{2,2})$. One randomly chooses $(S_i)_i, d, e \xleftarrow{\$} \mathbb{Z}_p$ to generate the server public keys $\mathsf{VK} = (g_i = g^{S_i})_i$, and $\mathsf{EK} = h = g^d$, and $\ell = g^e$ to set the round parameters $\mathsf{Rparam} = (\mathfrak{g}_r = \mathfrak{g}, \mathfrak{s}_r = \mathfrak{g}_r^{S_3})$.
For each honest user i, one randomly chooses $u_i, v_i, w_i, x_i, y_i, \mu_i, r_i \xleftarrow{\$} \mathbb{Z}_p$ to generate $\mathsf{vk}_i = (\mathfrak{g}_r; \mathfrak{f}_i = \mathfrak{g}_r^{u_i}, \mathfrak{l}_i = \mathfrak{g}_r^{y_i}, \mathfrak{g}_i = \mathfrak{g}_r^{v_i}, \mathfrak{h}_i = \mathfrak{g}_r^{x_i}; \mathfrak{A}_i = \mathfrak{g}_r^{w_i}, \mathfrak{B}_i = \mathfrak{A}_i^{w_i})$, and the signature Σ_i of vk_i under SK, $\pi_i = (\mathsf{Com}_i, \mathsf{Proof}_i)$, as well as $C_i = (g, \ell_i = \ell^{r_i}, a_i = g^{r_i}, b_i = h^{r_i} M_i)$, $C_0 = (1, \ell, g, h)$, with σ_i and τ_i the signatures of respectively C_i and C_0 under sk_i with respect to $\mathfrak{g}_r$. Indeed, knowing the signing keys allows to honestly generate the signatures Σ_i, σ_i, and τ_i. $\mathcal{B}_i = (C_i, \sigma_i, \tau_i, \mathsf{vk}_i, \Sigma_i, \pi_i)$ constitutes a ballot of an honest user. For the corrupted ones, the simulator receives directly $\mathcal{B}_i = (C_i, \sigma_i, \tau_i, \mathsf{vk}_i, \Sigma_i, \pi_i)$. The input ballot-box is then $\mathcal{BB}\mathsf{ox} = \{\mathcal{B}_i\}_i \cup \{(\mathsf{vk}_r, \overline{C_0})\}$ including the ballots of all the honest and corrupted users and the round parameters.
By randomly choosing α and γ_i, δ_i, z_i in $\mathbb{Z}_p$, the simulator also generates, as the mix-server would do, $\mathsf{Rparam}' = (\mathfrak{g}'_r = \mathfrak{g}_r^{\alpha}, \mathfrak{s}'_r = \mathfrak{s}_r^{\alpha})$ with a proof of knowledge of α such that $\mathfrak{g}'_r = \mathfrak{g}_r^{\alpha}$, and

$$\mathsf{vk}'_i = (\mathfrak{g}'_r; \mathfrak{f}'_i = \mathfrak{f}_i^{\alpha}, \mathfrak{l}'_i = \mathfrak{l}_i^{\alpha}\mathfrak{g}_r'^{\delta_i}, \mathfrak{g}'_i = \mathfrak{g}_i^{\alpha}, \mathfrak{h}'_i = \mathfrak{h}_i^{\alpha}; \mathfrak{A}'_i = \mathfrak{A}_i^{\alpha}, \mathfrak{B}'_i = \mathfrak{B}_i^{\alpha})$$
$$C'_i = (g, \ell'_i = \ell_i \cdot \ell^{\gamma_i}, a'_i = a_i \cdot g^{\gamma_i}, b'_i = b_i \cdot h^{\gamma_i})$$

$$\Sigma'_i = \Sigma_i^{\alpha} \cdot \mathfrak{s}_r'^{\delta_i} \qquad \sigma'_i = \sigma_i \cdot \tau_i^{\gamma_i} \cdot \ell_i'^{\delta_i} \qquad \tau'_i = \tau_i \cdot \ell_r'^{\delta_i}$$
$$c'_i = c_i \cdot v_{1,1}^{z_i} \qquad d'_i = d_i \cdot v_{1,2}^{z_i} \qquad \Theta'_i = (\Theta_i \cdot \mathfrak{g}_r^{z_i})^{\alpha} \qquad \Psi'_i = (\Psi_i \cdot \mathfrak{A}_i^{z_i})^{\alpha}$$

Eventually, the simulator chooses a random permutation Π on $\{1, \ldots, n\}$, and sets the output ballots $\mathcal{B}'_i = (\overline{C}'_{\Pi(i)}, \sigma'_{\Pi(i)}, \tau'_{\Pi(i)}, \mathsf{vk}'_{\Pi(i)}, \Sigma'_{\Pi(i)}, \pi'_{\Pi(i)})$. It outputs $\mathcal{BB}\mathsf{ox} = \{\mathcal{B}_i\}_i \cup \{(\mathsf{vk}_r, \overline{C}_0)\}$ and $\mathcal{BB}\mathsf{ox}' = \{\mathcal{B}'_i\} \cup \{(\mathsf{vk}'_r, \overline{C}_0)\}$, together with a zero-knowledge proof π' of Diffie-Hellman tuple for $(\mathfrak{g}_r, \mathfrak{g}'_r, \prod_i \mathfrak{A}_i, \prod_i \mathfrak{A}'_i)$.

Game $\mathbf{G}_1$: Our first step is to modify the CRS for the proofs of Square Diffie-Hellman tuples. To this aim, we change the global parameters, with a non-Diffie-Hellman tuple $(v_{1,1}, v_{1,2}, v_{2,1}, v_{2,2})$, so that $(v_{1,1}, v_{1,2}, v_{2,1}, g \cdot v_{2,2})$ is a Diffie-Hellman tuple, making the Groth-Sahai commitments perfectly hiding. Under the DDH assumption in $\mathbb{G}_1$, this makes no difference.

Game $\mathbf{G}_2$: Now, we explicitly set $(v_{1,1}, v_{1,2}, v_{2,1} = v_{1,1}^{\rho}, v_{2,2} = v_{1,2}^{\rho}/g)$ with a random known scalar ρ, so that $(v_{1,1}, v_{1,2}, v_{2,1}, g \cdot v_{2,2})$ is a Diffie-Hellman tuple. Thus, the Groth-Sahai commitments is perfectly hiding. For all the user, we can indeed see the commitment of w_i with randomness μ_i as $(c_i, d_i) = (v_{1,1}^{\mu_i + \rho w_i}, v_{1,2}^{\mu_i + \rho w_i})$, the commitment of 0 with randomness $\nu_i = \mu_i + \rho w_i$. Hence, $\Theta_i = \mathfrak{g}_r^{\mu_i} = \mathfrak{g}_r^{\nu_i - \rho w_i} = \mathfrak{A}_i^{-\rho} \cdot \mathfrak{g}_r^{\nu_i}$ and similarly $\Psi_i = \mathfrak{A}_i^{\mu_i} = \mathfrak{A}_i^{\nu_i - \rho w_i} = \mathfrak{B}_i^{-\rho} \cdot \mathfrak{A}_i^{\nu_i}$.
As for the proof π_i in each ballot, we can see the commitment in the proof π' by a commitment of 0 and simulate Θ, Ψ.

Game $\mathbf{G}_3$: Now, the public keys are generated as

$$\mathsf{vk}_i = (\mathfrak{g}_r; \mathfrak{f}_i = \mathfrak{g}_r^{u_i}, \mathfrak{l}_i = \mathfrak{g}_r^{v_i}, \mathfrak{g}_i = \mathfrak{g}_r^{x_i}, \mathfrak{h}_i = \mathfrak{g}_r^{y_i}; \mathfrak{A}_i = \mathfrak{g}_r^{w_i}, \mathfrak{B}_i = \mathfrak{g}_r^{\bar{w}_i})$$
$$\mathsf{vk}'_i = (\mathfrak{g}'_r = \mathfrak{g}_r^{\alpha}; \mathfrak{f}'_i = \mathfrak{f}_i^{\alpha}, \mathfrak{l}'_i = \mathfrak{l}_i^{\alpha} \mathfrak{g}_r'^{\delta_i}, \mathfrak{g}'_i = \mathfrak{g}_i^{\alpha}, \mathfrak{h}'_i = \mathfrak{h}_i^{\alpha}; \mathfrak{A}'_i = \mathfrak{A}_i^{\alpha}, \mathfrak{B}'_i = \mathfrak{B}_i^{\alpha})$$

from the known scalars $u_i, v_i, x_i, y_i, w_i, \bar{w}_i, \delta_i, \alpha$. Everything else can be simulated as above.

Lemma 20. *Under the* DSDH *assumption in $\mathbb{G}_2$, this game is indistinguishable from the previous one.*

Game $\mathbf{G}_4$: Notice that $\mathfrak{A}'_i = \mathfrak{A}_i^{\alpha} = \mathfrak{g}_r^{w_i \alpha} = \mathfrak{g}_r'^{w_i}$ and $\mathfrak{B}'_i = \mathfrak{B}_i^{\alpha} = \mathfrak{g}_r^{\bar{w}_i \alpha} = \mathfrak{g}_r'^{\bar{w}_i}$: they can be simulated without knowing α. And from the modification of the commitments applied with z_i, we can also set $\Theta'_i = \mathfrak{A}_i'^{-\rho} \cdot \mathfrak{g}_r'^{\nu'_i}$ and $\Psi'_i = \mathfrak{B}_i'^{-\rho} \cdot \mathfrak{A}_i'^{\nu'_i}$ with $\nu'_i = \nu_i + z_i$. Again, we can do the same in π' to simulate Θ', Ψ'. This game is perfectly indistinguishable from the previous one.

Game $\mathbf{G}_5$: For all the honest users, we replace C_i by an encryption of 1. And, even if we still choose at random the signing keys $\mathsf{sk}_i = (u_i, v_i, x_i, y_i; w_i, \bar{w}_i)$, we also choose at random $\mathsf{sk}'_i = (u'_i, v'_i, x'_i, y'_i)$.
This games is computationally indistinguishable from the previous one by the DDH assumption in $\mathbb{G}_1$ and $\mathbb{G}_2$ and the Unlinkability Assumption (see the two lemmas 15 and 16, in the proof Section 5.

Game $\mathbf{G}_6$: Finally, for all the user, we choose at random $w'_i, \bar{w}'_i$ in order to have random verification keys vk'_i. Under the DDH assumption in $\mathbb{G}_2$, this game is computationally indistinguishable from the previous one.

In this final game, vk'_i is a random key, for a secret key $\mathsf{sk}'_i = (u'_i, v'_i, x'_i, y'_i; w'_i, \bar{w}'_i) \xleftarrow{\$} \mathbb{Z}_p^6$, independent from the secret key $\mathsf{sk}_i = (u_i, v_i, x_i, y_i; w_i, \bar{w}_i)$. The ciphertexts C'_i contain random independent elements:

Proposition 21 (Unlinkability of the Mixing). *Given any input ballot-box and transformed ballot-box (following the protocol from Figure 5), for any pair of honestly generated input ballots $(\mathcal{B}_0, \mathcal{B}_1)$ and pair of output ballots $(\mathcal{B}'_0, \mathcal{B}'_1)$, such that $\mathcal{B}'_b$ is a randomization of $\mathcal{B}_0$, and $\mathcal{B}'_{1-b}$ is a randomization of $\mathcal{B}_1$, for a random bit $b \xleftarrow{\$} \{0, 1\}$, no adversary can get a non-negligible advantage in guessing b.*

Proof (Lemma 20). The proof works with a sequence of hybrid games:

Game $\mathbf{G}_0$: Since we do not need w_i anymore for the proofs, the simulator receives a random Square Diffie-Hellman tuple $(\mathfrak{g}_r, \mathfrak{A}_i, \mathfrak{B}_i)$ for all the users, and everything else can be simulated as above, without w_i but with random ν_i, ν'_i and using ρ. In particular, the public keys can be generated as

$$\mathsf{vk}_i = (\mathfrak{g}_r; \mathfrak{f}_i = \mathfrak{g}_r^{u_i}, \mathfrak{l}_i = \mathfrak{g}_r^{v_i}, \mathfrak{g}_i = \mathfrak{g}_r^{x_i}, \mathfrak{h}_i = \mathfrak{g}_r^{y_i}; \mathfrak{A}_i, \mathfrak{B}_i)$$
$$\mathsf{vk}'_i = (\mathfrak{g}'_r = \mathfrak{g}_r^{\alpha}; \mathfrak{f}'_i = \mathfrak{f}_i^{\alpha}, \mathfrak{l}'_i = \mathfrak{l}_i^{\alpha}\mathfrak{g}_r'^{\delta_i}, \mathfrak{g}'_i = \mathfrak{g}_i^{\alpha}, \mathfrak{h}'_i = \mathfrak{h}_i^{\alpha}; \mathfrak{A}'_i = \mathfrak{A}_i^{\alpha}, \mathfrak{B}'_i = \mathfrak{B}_i^{\alpha})$$

from the known scalars $u_i, v_i, x_i, y_i, \delta_i$.

Game $\mathbf{G}_1$: The simulator now receives random tuples $(\mathfrak{g}_r, \mathfrak{A}_i, \mathfrak{B}_i)$ and runs as before. This game is indistinguishable from the previous one thanks to the Square Diffie-Hellman assumption.

Game $\mathbf{G}_2$: We now do the simulation with

$$\mathsf{vk}_i = (\mathfrak{g}_r; \mathfrak{f}_i = \mathfrak{g}_r^{u_i}, \mathfrak{l}_i = \mathfrak{g}_r^{v_i}, \mathfrak{g}_i = \mathfrak{g}_r^{x_i}, \mathfrak{h}_i = \mathfrak{g}_r^{y_i}; \mathfrak{A}_i = \mathfrak{g}_r^{w_i}, \mathfrak{B}_i = \mathfrak{g}_r^{\bar{w}_i})$$
$$\mathsf{vk}'_i = (\mathfrak{g}'_r = \mathfrak{g}_r^{\alpha}; \mathfrak{f}'_i = \mathfrak{f}_i^{\alpha}, \mathfrak{l}'_i = \mathfrak{l}_i^{\alpha}\mathfrak{g}_r'^{\delta_i}, \mathfrak{g}'_i = \mathfrak{g}_i^{\alpha}, \mathfrak{h}'_i = \mathfrak{h}_i^{\alpha}; \mathfrak{A}'_i = \mathfrak{A}_i^{\alpha}, \mathfrak{B}'_i = \mathfrak{B}_i^{\alpha})$$

for known random scalars $u_i, v_i, x_i, y_i, w_i, \bar{w}_i, \delta_i, \alpha \xleftarrow{\$} \mathbb{Z}_p$. This game is perfectly indistinguishable from the previous one. □

B.5 Generation of the Parameters and Keys

As already explained, for the soundness and the zero-knowledge property to hold in the above mixing protocol, we need w_i unknown to anybody (if one wants to consider possible collusion between the mix-server and some senders). For electronic voting, this means that the voters can choose (u_i, v_i, x_i, y_i) but should jointly generate $(\mathfrak{A}_i, \mathfrak{B}_i)$ together with the proof π_i such that nobody knows w_i, to build vk_i and get the signature Σ_i.

Of course, this could be done with generic two-party computation, between the user and the signing authority, but we present an efficient generation of the vk_i and the signature Σ_i.

– The voter chooses $w_{i,1} \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$(\mathfrak{A}_{i,1} = \mathfrak{g}_r^{w_{i,1}}, \mathfrak{B}_{i,1} = \mathfrak{A}_{i,1}^{w_{i,1}})$$

with a proof $\pi_{i,1}$ of Square Diffie-Hellman tuple for $(\mathfrak{g}_r, \mathfrak{A}_{i,1}, \mathfrak{B}_{i,1})$:

$$\pi_{i,1} = (\mathsf{Com}_{i,1}, \mathsf{Proof}_{i,1}) \text{ where}$$
$$\mathsf{Com}_{i,1} = (c_{i,1} = v_{2,1}^{w_{i,1}} v_{1,1}^{\mu_{i,1}}, d_{i,1} = v_{2,2}^{w_{i,1}} v_{1,2}^{\mu_{i,1}} g^{w_{i,1}})$$
$$\mathsf{Proof}_{i,1} = (\Theta_{i,1} = \mathfrak{g}_r^{\mu_{i,1}}, \Psi_{i,1} = \mathfrak{A}_{i,1}^{\mu_{i,1}}).$$

The voter also chooses $u_i, v_i, x_i, y_i \xleftarrow{\$} \mathbb{Z}_p$ and sends $(\mathfrak{f}_i = \mathfrak{g}_r^{u_i}, \mathfrak{l}_i = \mathfrak{g}_r^{v_i}, \mathfrak{g}_i = \mathfrak{g}_r^{x_i}, \mathfrak{h}_i = \mathfrak{g}_r^{y_i}; \mathfrak{A}_{i,1}, \mathfrak{B}_{i,1}, \pi_{i,1})$ to the signer.

– On its side, the signer chooses $w_{i,2} \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$\mathfrak{A}_{i,2} = \mathfrak{g}_r^{w_{i,2}} \qquad \mathfrak{B}_{i,2} = (\mathfrak{A}_{i,1}^2 \cdot \mathfrak{g}_r^{w_{i,2}})^{w_{i,2}}$$

with its contribution $\pi_{i,2}$ for a proof π_i of Square Diffie-Hellman tuple for $(\mathfrak{g}_r, \mathfrak{A}_i = \mathfrak{A}_{i,1} \cdot \mathfrak{A}_{i,2}, \mathfrak{B}_i = \mathfrak{B}_{i,1} \cdot \mathfrak{B}_{i,2})$:

$$\pi_{i,2} = (\mathsf{Com}_{i,2}, \mathsf{Proof}_{i,2}) \text{ where}$$
$$\mathsf{Com}_{i,2} = (c_{i,2} = v_{2,1}^{w_{i,2}} v_{1,1}^{\mu_{i,2}}, d_{i,2} = v_{2,2}^{w_{i,2}} v_{1,2}^{\mu_{i,2}} g^{w_{i,2}})$$
$$\mathsf{Proof}_{i,1} = (\Theta_{i,2} = \mathfrak{g}_r^{\mu_{i,2}}, \Psi_{i,2} = \Theta_{i,1}^{w_{i,2}} \cdot (\mathfrak{A}_{i,1}\mathfrak{A}_{i,2})^{\mu_{i,2}})$$

Then, the signer publishes

$$\mathfrak{A}_i = \mathfrak{A}_{i,1} \cdot \mathfrak{A}_{i,2} = \mathfrak{g}_r^{w_{i,1}+w_{i,2}}$$
$$\mathfrak{B}_i = \mathfrak{B}_{i,1} \cdot \mathfrak{B}_{i,2} = \mathfrak{A}_{i,1}^{w_{i,1}} \cdot \mathfrak{A}_{i,1}^{2w_{i,2}} \mathfrak{g}_r^{w_{i,2}^2} = \mathfrak{g}_r^{w_{i,1}^2+2w_{i,1}w_{i,2}+w_{i,2}^2} = \mathfrak{g}_r^{(w_{i,1}+w_{i,2})^2}$$

together with the proof

$$\begin{aligned} c_i &= c_{i,1} \cdot c_{i,2} = v_{2,1}^{w_{i,1}+w_{i,2}} v_{1,1}^{\mu_{i,1}+\mu_{i,2}} \\ d_i &= d_{i,1} \cdot d_{i,2} = v_{2,2}^{w_{i,1}+w_{i,2}} v_{1,2}^{\mu_{i,1}+\mu_{i,2}} g^{w_{i,1}+w_{i,2}} \\ \Theta_i &= \Theta_{i,1} \cdot \Theta_{i,2} = \mathfrak{g}_r^{\mu_{i,1}+\mu_{i,2}} \\ \Psi_i &= \Psi_{i,1} \cdot \Psi_{i,2} = \mathfrak{A}_{i,1}^{\mu_{i,1}} \cdot \Theta_{i,1}^{w_{i,2}} \cdot (\mathfrak{A}_{i,1}\mathfrak{A}_{i,2})^{\mu_{i,2}} \\ &= \mathfrak{A}_{i,1}^{\mu_{i,1}} \cdot \mathfrak{A}_{i,2}^{\mu_{i,1}} \cdot (\mathfrak{A}_{i,1}\mathfrak{A}_{i,2})^{\mu_{i,2}} = \mathfrak{A}_i^{\mu_{i,1}+\mu_{i,2}}. \end{aligned}$$

The signer also generates the signature $\Sigma_i = \mathfrak{g}_r^{S_1} \mathfrak{f}_i^{S_2} \mathfrak{l}_i^{S_3} \mathfrak{g}_i^{S_4} \mathfrak{h}_i^{S_5} \mathfrak{A}_i^{S_6} \mathfrak{B}_i^{S_7}$, together with an interactive zero-knowledge proof of knowledge of $w_{i,2}$ such that $\mathfrak{A}_{i,2} = \mathfrak{A}_i/\mathfrak{A}_{i,1} = \mathfrak{g}_r^{w_{i,2}}$.

The signature Σ_i can only be used by the voter who knows $\mathsf{sk}_i = (u_i, v_i, x_i, y_i)$ for the verification key $\mathsf{vk}_i = (\mathfrak{f}_i, \mathfrak{l}_i, \mathfrak{g}_i, \mathfrak{h}_i; \mathfrak{A}_i, \mathfrak{B}_i)$. But the proof of knowledge of $w_{i,2}$ guarantees that w_i is really jointly generated. Indeed, in order to prove that the tuple $(\mathfrak{g}_r, \mathfrak{A}_i, \mathfrak{B}_i)$ is random, we can do the simulation of the above protocol with respect to the users and the mix-server, by simulating the signer:

- first, we replace the setup of the Groth-Sahai commitments to make them perfectly hiding and to allow simulation of the GS proofs;
- we also use the simulation of the interactive zero-knwoledge proof of knowledge of $w_{i,2}$ such that $\mathfrak{A}_{i,2} = \mathfrak{A}_i/\mathfrak{A}_{i,1} = \mathfrak{g}_r^{w_{i,2}}$;
- we now receive a random tuple $(\mathfrak{g}_r, \mathfrak{A}_i, \mathfrak{B}_i)$, for unknown w_i: we can do as before with $\mathfrak{A}_{i,2} = \mathfrak{A}_i/\mathfrak{A}_{i,1}$ and $\mathfrak{B}_{i,2} = \mathfrak{B}_i/\mathfrak{B}_{i,1}$, as the proofs are simulated.