Extractable Linearly-Homomorphic Signatures and Scalable Mix-Nets

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. It is also linear in the number of ciphertexts, independently of the number of rounds of mixing, for the verifiers. This leads to the most efficient technique, that is highly scalable.

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 brick to construct a set of anonymous 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].

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 remains 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. But with N multiple mixing steps, which are required if one wants to guarantee anonymity even if some mix-servers are malicious, the communication is linear in N, 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]. However, 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.

We propose a totally new approach that can treat each ballot in an independent way, with just a constant-size overhead. In addition, the overheads for each shuffle can be updated, to still keep it constant-size, independently of the number of rounds of mixing. From our knowledge, this is the most efficient and scalable solution.

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 output ballot-box, it is a randomization of some $C_{\Pi(i)}$ from the input ballot-box (surjectivity) and vice-versa (injectivity). Then, the output ciphertexts can be mixed again by another mix-server, hence the notion of mix-network.

Our approach to avoid the proof of an explicit permutation on all the ciphertexts but with still the appropriate properties deeply uses 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 (multiplication by a constant). Unfortunately, linearly-homomorphic signature schemes not only allow multiplication by a constant, but also linear combinations, which would allow combinations of ballots. In order to avoid such combinations, we propose a new kind of *non-miscibility* notion, with Square Diffie-Hellman tuples $(g, g^{x_i}, g^{x_i^2})$. It limits signatures to multiplications by constant values only. These two properties together essentially guarantee the permutation property.

– unlinkability will be satisfied thanks to the randomizations that are indistinguishable for various users, under some DDH-like assumptions.

With the above technique of combining linearly-homomorphic signatures on messages containing Square Diffie-Hellman tuples, we indeed guarantee that the output C'_i is a randomization of an input C_i, while the randomization also makes them unlinkable. One will have to make sure that there is no duplicates and the same numbers of ballots in the input ballot-box and output ballot-box for the formal proof of permutation.

Nevertheless, this technique of randomizing ciphertexts and verification keys and adapting signatures can be seen as an extension of signatures on randomizable ciphertexts [BFPV11] which did not allow updates of the verification keys. This excluded anonymity. Our new approach can find several applications, as already illustrated in this paper to anonymous credentials (see Section 4) and electronic voting (see Section 7), 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 an assumption in the generic 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. In our case, we assume instead the existence of an extractor for linear combinations of signatures, which is close to the extractability assumptions used in SNARKs [GW11, BCCT12, GGPR13]. They are both proven in the generic group model, but it allows us 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 9 exponentiations in $\mathbb{G}_1$ and 13 exponentiations in $\mathbb{G}_2$ per ciphertext plus an overhead of 3 exponentiations in $\mathbb{G}_1$ and 2 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.

Eventually, they have a common reference string that is linear in the number of ciphertexts and argue that "One could wish for a common reference string that has only a constant number of group elements, but currently even all known

3-move zero-knowledge arguments have common reference strings of size $\Omega(n)$". In the CRS, one needs to randomly pick $x_i \xleftarrow{\$} \mathbb{Z}_p$ for all the ciphertexts and compute $g^{x_i}, g^{x_i^2}$. Nobody does not learn the x_is. We solve the raised issue within our global protocol, with a constant-size common reference string: it just contains bilinear group setting and a non-Diffie-Hellman tuple for Groth-Sahai proofs.

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 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. We also introduce our second core assumption (the *extractability assumption*), that also holds in the generic group model. We then apply these two assumptions to traceable anonymous credentials (Section 4) and mix-networks (Section 5), followed by a detailed security analysis in Section 6. Eventually, we explain application to electronic voting in Section 7.

2 Computational Assumptions

In this section, we will first recall some classical computational assumptions and introduce a new one, of independent interest, as it opens the floor for new anonymous credentials and a class of privacy-preserving protocols.

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

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

Definition 3 (Decisional Diffie-Hellman Assumption (DDH)). *The Decisional Diffie-Hellman (DDH) Assumption in a group $\mathbb{G}$ of prime order p 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 even 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 4 (Decisional Square Diffie-Hellman Assumption (DSDH)). *The Decisional Square Diffie-Hellman (DSDH) Assumption in a group $\mathbb{G}$ of prime order p 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 $h \in \mathbb{G}_1$, with $g \in \mathbb{G}_1$, $\mathfrak{g} \in \mathbb{G}_2$, $r, t \in \mathbb{Z}_p$:

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

Definition 5 (Unlinkability Assumption). *The Unlinkability Assumption states that for any $g, h \in \mathbb{G}_1$ and $\mathfrak{g} \in \mathbb{G}_2$, with the definition below, the distributions $\mathcal{D}_{h,g,\mathfrak{g}}(u, u)$ and $\mathcal{D}_{h,g,\mathfrak{g}}(u, v)$ are computationally indistinguishable, for any $u, v \xleftarrow{\$} \mathbb{Z}_p$:*

$$\mathcal{D}_{h,g,\mathfrak{g}}(u, v) = \left\{ (\mathsf{Cred}(u, h; g, \mathfrak{g}, r, t), \mathsf{Cred}(v, h; g', \mathfrak{g}', r', t')); \begin{array}{l} g' \xleftarrow{\$} \mathbb{G}_1, \mathfrak{g}' \xleftarrow{\$} \mathbb{G}_2, \\ r, t, r', t' \xleftarrow{\$} \mathbb{Z}_p \end{array} \right\}$$

Intuitively, the third component is an ElGamal ciphertext of the fixed value h^u, which hides it, and makes indistinguishable another encryption h^u from an encryption of h^v while, given $(\mathfrak{g}, \mathfrak{g}^u)$, $\mathfrak{g}'^u$ and $\mathfrak{g}'^v$ are indistinguishable for random $\mathfrak{g}' \xleftarrow{\$} \mathbb{G}_2$, under the DDH assumption in $\mathbb{G}_2$. However the pairing relation allows to check consistency:

$$e(g^{rt} \cdot h^u, \mathfrak{g}) = e(g^r, \mathfrak{g}^t) \cdot e(h, \mathfrak{g}^u) = e(g^r, \mathfrak{g}^t) \cdot e(h^u, \mathfrak{g}).$$

Because of the independent group elements in the two credentials, this assumption clearly holds in the generic bilinear group model.

Thanks to this unlinkability assumption, and the randomizability of the above credential, proving knowledge of u can lead to anonymous credentials, as explained in Section 4.

3 Linearly Homomorphic Signatures

The notion of homomorphic signatures dates back to [JMSW02], with various 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 are 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 $\mathfrak{g}_u, \mathfrak{g}_v, \mathfrak{h}_u, \mathfrak{h}_w \xleftarrow{\$} \mathbb{G}_2$, 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

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

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

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

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

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 $(M_i)_i$ and $(M'_i)_i$, 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 $M'' = ({M_i}^{\alpha}{M''_i}^{\beta})_i$.

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 $(M_i)_i \in \mathbb{G}_1^n$, one sets $\sigma = \prod_{i=1}^n M_i^{s_i}$;

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

As above, this signature scheme is linearly homomorphic: 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 $(M_i)_i$ and $(M'_i)_i$, with their signatures σ and σ', then $\sigma'' = \sigma^{\alpha}{\sigma''}^{\beta}$ is a valid signature of $M'' = ({M_i}^{\alpha}{M''_i}^{\beta})_i$.

We now show that any new valid pair $((M_i)_i, \sigma)$ produced by an adversary, in the generic group model, is necessarily for a linear combination of the already signed vector-messages, and an extractor can provide the coefficients applied by the adversary.

3.3 Extractability of Linear Combinations

First, we consider adversaries in the generic bilinear group model [BBG05]. To prove 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. Then, we will state an assumption in the standard model: for any adversary that generates a new valid pair $(\boldsymbol{M}, \sigma)$ from a list of valid pairs $(\boldsymbol{M}_i, \sigma_i)_i$, all under the same verification key, there exists an extractor that outputs the scalars $(\alpha_i)_i$ such that $\boldsymbol{M} = \prod_i \boldsymbol{M}_i^{\alpha_i}$.

We stress that the theorem below is proven in the generic bilinear group model, which means that an adversary can only combine group elements through the addition oracles (in $\mathbb{G}_1$, $\mathbb{G}_2$ or $\mathbb{G}_T$) and the pairing oracle, to generate a new group element. The assumption we will later make will apply to any adversary in the standard model.

Theorem 6. *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, one also gets $(\alpha_j)_j$ such that $\boldsymbol{M} = \prod_j \boldsymbol{M}_j^{\alpha_j}$ and $\sigma = \prod_j \sigma_j^{\alpha_j}$.*

Proof. The adversary is given $((M_{j,i})_i, \sigma_j)_j$ 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:

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

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:

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

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 $\sigma = \prod_j \sigma_j^{\delta_j}$. □

We can now state our computational assumption that holds in the generic bilinear group model:

Definition 7 (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 $\sigma = \prod_j \sigma_j^{\alpha_j}$.*

3.4 Non-Miscibility

Once we know only linear combinations are possible among signed vector-messages, one may require stronger restrictions such as avoiding combinations between different vectors, in order to limit to the sole multiplication of a vector by a constant, as a kind of randomization. We show a particular case of non-miscibility of vectors: from multiple distinct Square Diffie-Hellman tuples $(g_i, g_i^{x_i}, g_i^{x_i^2})$, 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 x_i's (it will be applied in our anonymous credentials, Section 4), or with a common basis $g_i = g$, but secret x_i's (it will be applied in our mix-net, Section 5). More precisely:

Theorem 8. *Given n valid Square Diffie-Hellman tuples $(g_i, a_i = g_i^{x_i}, b_i = a_i^{x_i})$, with x_i, for random $g_i \xleftarrow{\$} \mathbb{G}$ and $x_i \xleftarrow{\$} \mathbb{Z}_p$, 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.*

Proof. We stress that in the above 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 $\alpha_1 K u + \alpha_2(V - 2Ux_2 + Tx_2^2)v + (TV - U^2) = 0 \bmod p$, which means that the discrete logarithm of Z in the basis g is $u = -(\alpha_2(V - 2Ux_2 + Tx_2^2)v + (TV - U^2))/(\alpha_1 K) \bmod p$. □

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

Theorem 9. *Given 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 $x_i \xleftarrow{\$} \mathbb{Z}_p$, 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.*

Proof. We first prove the following lemma.

Lemma 10. *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 10 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.

□

4 Traceable Anonymous Credentials

As a first application, and a warm-up, we will explain how our new unlinkability assumption (see Section 2.2) and non-miscibility (see Section 3.4) can be used for anonymous credentials, allowing public traceability.

4.1 Construction

On Figure 1, we provide a full description of the setup, the credential issuing procedure, the credential proving protocol, and the tracing procedure. One can note that only the owner of SK, associated to VK (and thus the Credential Authority), can generate signatures $\Sigma_{i,j}$, which satisfy

$$e(g, \Sigma_{i,j}) = e(g_1, \mathfrak{g}) \cdot e(g_2, \mathfrak{k}_i) \cdot e(g_3, \mathfrak{A}_i) \cdot e(g_4, \mathfrak{B}_i) \cdot e(h_j, \mathfrak{U}_{i,j}).$$

When using this credential, the user $\mathcal{X}_i$ introduces a randomness T_j:

$$e(g, \Sigma_{i,j}^{T_j}) = e(g_1, \mathfrak{g}^{T_j}) \cdot e(g_2, \mathfrak{k}_i^{T_j}) \cdot e(g_3, \mathfrak{A}_i^{T_j}) \cdot e(g_4, \mathfrak{B}_i^{T_j}) \cdot e(h_j, \mathfrak{U}'_{i,j}).$$

Hence, the multiplication leads to, with $T = \sum_{\mathcal{J}} T_j$ and $\Sigma = \prod_{\mathcal{J}} \Sigma_{i,j}$,

$$e(g, \Sigma) = e(g_1, \mathfrak{g}^T) \cdot e(g_2, \mathfrak{k}_i^T) \cdot e(g_3, \mathfrak{A}_i^T) \cdot e(g_4, \mathfrak{B}_i^T) \cdot \prod_{\mathcal{J}} e(h_j, \mathfrak{U}'_{i,j}).$$

We will show below that the various executions of the credential proving protocol are unlinkable, which guarantees strong anonymity for the users. Indeed, the various $\mathfrak{U}'_{i,j}$ are unlinkable, under the DDH assumption in $\mathbb{G}_2$. However, once some $h_{i,j} = H_t^{u_{i,j}}$ is revealed, its use in a credential $\mathsf{Cred}(u_{i,j}, H_t; g', \mathfrak{g}', r', s')$ can be detected.

We stress that our construction is not the most efficient, even compared to the anonymous credentials proposed by Camenish and Lysyanskaya [CL04]. But our new general approach is innovative, and a good warm-up for our mix-net protocol. It also accepts an efficient public tracing procedure with forward and backward pricacy.

4.2 Security Analysis

Unforgeability. This property requires that no adversary can successfully run the credential proving protocol for an attribute $\mathcal{A}_j$, without having obtained the credential earlier from the Credential Authority. Because of the proof of knowledge of x_i such that $(\mathfrak{k}, \mathfrak{A} = \mathfrak{k}^{x_i}, \mathfrak{B} = \mathfrak{A}^{x_i})$, this proves the Square Diffie-Hellman property. Theorem 8 ensures this tuple corresponds to a tuple $(\mathfrak{k}_i, \mathfrak{A}_i, \mathfrak{B}_i)$ associated to a unique $\mathcal{X}_i$, even if several users knowing their x_i's collude. Hence, the signature Σ guarantees that it comes from linear combinations of $(\mathfrak{g}, \mathfrak{k}_i, \mathfrak{A}_i, \mathfrak{B}_i, \mathfrak{U}_{i,j})$ for various j's, including the index of $\mathcal{A}\mathsf{ttr}_j$, but for a unique $\mathcal{X}_i$. Hence, $\mathcal{X}_i$ has been granted the attribute $\mathcal{A}\mathsf{ttr}_j$.

Initialization

- 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, then the global parameters are $\mathsf{Gparam} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, \mathfrak{g}, e)$;
- Credential Authority keys: the signing key $\mathsf{SK} = (s_1, s_2, s_3, s_4, (S_i)_i) \xleftarrow{\$} \mathbb{Z}_p^{4+n}$, and the associated verification key $\mathsf{VK} = ((g_i = g^{s_i})_i, (h_j = g^{S_j})_j)$. Each key h_j will be associated to an attribute $\mathcal{A}\mathsf{ttr}_j$;
- Time-period parameter: $H_t = h$, where $h \xleftarrow{\$} \mathbb{G}_1$ is a random generator. We will identify time-periods by H_t;
- User secret: each user receives a random generator $\mathfrak{k}_i \xleftarrow{\$} \mathbb{G}_2$ from the Tracing Authority, chooses a random scalar $x_i \xleftarrow{\$} \mathbb{Z}_p$, and sends back $\mathcal{X}_i = g^{x_i}$ to the Tracing Authority that stores $(\mathcal{X}_i, \mathfrak{k}_i)$.
  We will identify users by $\mathcal{X}_i$. User $\mathcal{X}_i$ stores $(x_i, \mathfrak{k}_i)$ which defines the Square-Diffie-Hellman tuple $(\mathfrak{k}_i, \mathfrak{A}_i = \mathfrak{k}_i^{x_i}, \mathfrak{B}_i = \mathfrak{A}_i^{x_i})$.

Credential Issuing Protocol

For a time-period H_t, user $\mathcal{X}_i$ wants to obtain a credential $\mathcal{C}_{i,j}$ for the attribute $\mathcal{A}\mathsf{ttr}_j$:

- it chooses a random scalar $u_{i,j} \xleftarrow{\$} \mathbb{Z}_p$ and sets $\mathfrak{U}_{i,j} = \mathfrak{g}^{u_{i,j}}$;
- it sends $(\mathcal{X}_i = g^{x_i}, \mathfrak{k}_i, \mathfrak{A}_i = \mathfrak{k}_i^{x_i}, H_t^{u_{i,j}}, \mathfrak{U}_{i,j} = \mathfrak{g}^{u_{i,j}})$ to the Tracing Authority, and runs two zero-knowledge proofs of Diffie-Hellman tuples for $(g, \mathcal{X}_i = g^{x_i}, \mathfrak{k}_i, \mathfrak{A}_i = \mathfrak{k}_i^{x_i})$ and $(H_t, H_t^{u_{i,j}}, \mathfrak{g}, \mathfrak{U}_{i,j} = \mathfrak{g}^{u_{i,j}})$. The Tracing Authority *certifies* the tuple $(\mathcal{X}_i, \mathfrak{k}_i, \mathfrak{A}_i, \mathfrak{U}_{i,j})$ for the Credential Authority, and stores $(\mathcal{X}_i, \mathfrak{k}_i, h_{i,j} = H_t^{u_{i,j}})$ for later tracing;
- it sends $C = \mathsf{Cred}(u_{i,j}, H_t; g, \mathfrak{g}, r, s)$, for random scalars $r, s \xleftarrow{\$} \mathbb{Z}_p$ to the Credential Authority together with $(\mathfrak{k}_i, \mathfrak{A}_i = \mathfrak{k}_i^{x_i}, \mathfrak{B}_i = \mathfrak{A}_i^{x_i})$ and the *certification* from the Tracing Authority on $(\mathcal{X}_i, \mathfrak{k}_i, \mathfrak{A}_i, \mathfrak{U}_{i,j})$. It also runs a zero-knowledge proof of knowledge of x_i such that $(\mathcal{X}_i = g^{x_i}, \mathfrak{k}_i, \mathfrak{A}_i = \mathfrak{k}_i^{x_i}, \mathfrak{B}_i = \mathfrak{A}_i^{x_i})$.

If convinced, the Credential Authority produces a signature on $(\mathfrak{g}, \mathfrak{k}_i, \mathfrak{A}_i, \mathfrak{B}_i, \mathfrak{U}_{i,j})$ as $\Sigma_{i,j} = \mathfrak{g}^{s_1} \mathfrak{k}_i^{s_2} \mathfrak{A}_i^{s_3} \mathfrak{B}_i^{s_4} \mathfrak{U}_{i,j}^{S_j}$. The user $\mathcal{X}_i$ stores $(x_i, \mathfrak{k}_i, (u_{i,j}, \Sigma_{i,j})_j)$ where j ranges on the validated attributes $\mathcal{A}\mathsf{ttr}_j$.

Credential Proving Protocol

For a time-period H_t, when a user $\mathcal{X}_i$ wants to prove to own credentials $\{\mathcal{A}\mathsf{ttr}_j, j \in \mathcal{J}\}$ for some subset $\mathcal{J}$ of attributes indices, it provides randomized individual credentials $\mathsf{Cred}(u_{i,j}, H_t; g^{R_j}, \mathfrak{h}_j = \mathfrak{g}^{T_j}, r_j, r'_j)$ for random scalars $r_j, r'_j, R_j, T_j \xleftarrow{\$} \mathbb{Z}_p$ (which include $\mathfrak{U}'_{i,j} = \mathfrak{U}_{i,j}^{T_j}$), as well as $\Sigma = \prod_{\mathcal{J}} \Sigma_{i,j}^{T_j}$ and, for $T = \sum_{\mathcal{J}} T_j$,$(\mathfrak{h} = \mathfrak{g}^T, \mathfrak{k} = \mathfrak{k}_i^T, \mathfrak{A} = \mathfrak{A}_i^T = \mathfrak{k}^{x_i}, \mathfrak{B} = \mathfrak{B}_i^T = \mathfrak{A}^{x_i})$. It then interactively runs a zero-knowledge proof of knowledge of x_i such that $(\mathfrak{k}, \mathfrak{A} = \mathfrak{k}^{x_i}, \mathfrak{B} = \mathfrak{A}^{x_i})$. The verifier checks whether

$$e(g, \Sigma) = e(g_1, \mathfrak{h}) \cdot e(g_2, \mathfrak{k}) \cdot e(g_3, \mathfrak{A}) \cdot e(g_4, \mathfrak{B}) \cdot \textstyle\prod_{\mathcal{J}} e(h_j, \mathfrak{U}'_{i,j}).$$

Tracing Procedure

Each time period, the Tracing Authority updates the time-period parameter $H'_t \leftarrow H_t^r$ for a new random scalar $r \xleftarrow{\$} \mathbb{Z}_p$, and also adapts its records $(\mathcal{X}_i, \mathfrak{k}_i, h_{i,j})$ into $(\mathcal{X}_i, \mathfrak{k}_i, h_{i,j}^r)$. To trace the attribute $\mathcal{A}_j$ for user $\mathcal{X}_i$ during this time-period H_t, the Tracing Authority can publish $h_{i,j} = H_t^{u_{i,j}}$ in the tracing list. Then anytime the service runs the Credential Proving Protocol, it first checks whether for some individual credential $\mathsf{Cred}(u_{i,j}, H_t; g', \mathfrak{g}', r', s')$: $e(g'^{r's'} \cdot H_t^{u_{i,j}}, \mathfrak{g}') = e(g'^{r'}, \mathfrak{g}'^{s'}) \cdot e(h^*, \mathfrak{g}')$ for some h^* in the tracing list. If yes, then it aborts.

Fig. 1. Anonymous Credentials

Unlinkability. Anonymous credentials guarantee that several uses of the same credential cannot be linked together, and thus, *a fortiori*, cannot be linked to the owner either.

This unlinkability should even be guaranteed with respect to the Credential Authority. We will thus simulate the view of the latter, including a credential issuing and its associated proving protocol, which we will prove indistinguishable to a view for two independent credential issuing and proving executions. The sequence below is a brief sketch of proof. Again, this anonymous credential is not the main part of this paper, but this proof gives the intuition of the next proof of the mix-net anonymity, which will be given in more details in Section 6.4. It starts from the real game, where the same scalars are used in the Credential Issuing Protocol and in the Credential Proving Protocol, and independent scalars are used in the last game. Indistinguishability between the games proves that issuing and proving protocol executions with similar scalars, and namely x_i, $u_{i,j}$'s (initial game) are indistinguishable from issuing and protocol executions with independent scalars, such as x_i, $u_{i,j}$'s vs. x'_i, $u'_{i,j}$'s (last game):

- the initial simulation is a perfect simulation with known SK, and honestly generated $(\mathfrak{k}_i, \mathfrak{A}_i = \mathfrak{k}_i^{x_i}, \mathfrak{A}_i^{x_i})$ and $C_{i,j} = \mathsf{Cred}(u_{i,j}, H_t; g, \mathfrak{g}, r, s)$, for random scalars $x_i, u_{i,j}, r, s \xleftarrow{\$} \mathbb{Z}_p$ in the credential issuing protocol, and $C'_{i,j} = \mathsf{Cred}(u_{i,j}, H_t; g^{R_j}, \mathfrak{g}^{T_j}, r_j, s_j)$, for random scalars $R_j, T_j, r_j, s_j \xleftarrow{\$} \mathbb{Z}_p$ in the credential proving protocol;
- the first modification consists in simulating all the zero-knowledge proofs;
- since no proofs are needed anymore, one can use random tuples $(\mathfrak{k}_i, \mathfrak{A}_i = \mathfrak{k}_i^{x_i}, \mathfrak{A}_i^{y_i})$, which is indistinguishable under the DSDH assumption in $\mathbb{G}_2$ (note that $\mathfrak{k}_i$ and x_i should be chosen by the honest players, either the Revocation Authority or the User);
- we can now generate theses tuples as $(\mathfrak{k}_i = \mathfrak{g}^{k_i}, \mathfrak{A}_i = \mathfrak{k}_i^{x_i}, \mathfrak{B}_i = \mathfrak{A}_i^{y_i})$, for random $k_i, x_i \xleftarrow{\$} \mathbb{Z}_p$;
- the credentials $(C_{i,j}, C'_{i,j})$ are generated from $\mathcal{D}_{H_t,g,\mathfrak{g}}(u_{i,j}, u_{i,j})$, without knowing the random scalars, since one can compute $(\mathfrak{h} = \prod \mathfrak{h}_j, \mathfrak{k} = \mathfrak{h}^{k_i}, \mathfrak{A} = \mathfrak{k}^{x_i}, \mathfrak{B} = \mathfrak{A}^{y_i})$, and generate the signature using SK;
- the credentials $(C_{i,j}, C'_{i,j})$ are now generated from $\mathcal{D}_{H_t,g,\mathfrak{g}}(u_{i,j}, u'_{i,j})$, for $u_{i,j}, u'_{i,j} \xleftarrow{\$} \mathbb{Z}_p$, which is indistinguishable, under the unlinkability assumption;
- we can now generate the credentials knowing the scalars, but still for independent $u_{i,j}$ and $u'_{i,j}$. Then, one can compute $(\mathfrak{h} = \mathfrak{g}^T, \mathfrak{k} = \mathfrak{k}_i^T, \mathfrak{A} = \mathfrak{A}_i^T, \mathfrak{B} = \mathfrak{B}_i^T)$;
- we are now given a multi Diffie-Hellman tuple $(\mathfrak{g}, \mathfrak{k}_i, \mathfrak{A}_i, \mathfrak{B}_i, \mathfrak{h} = \mathfrak{g}^T, \mathfrak{k} = \mathfrak{k}_i^T, \mathfrak{A} = \mathfrak{A}_i^T, \mathfrak{B} = \mathfrak{B}_i^T)$, from which we can generate the first credentials (in the Credential Proving Protocol) with random T_j's, but the last one (with index k) uses $\mathfrak{h}_k = \mathfrak{h} / \prod_j \mathfrak{h}_j$, where j ranges on all the previous indices;
- we are now given a random tuple $(\mathfrak{g}, \mathfrak{k}_i, \mathfrak{A}_i, \mathfrak{B}_i, \mathfrak{h}, \mathfrak{k}, \mathfrak{A}, \mathfrak{B})$, and do the same as above, which is indistinguishable under the DDH assumption in $\mathbb{G}_2$.

In this last game, the view of the verifier (which can possibly be the Credential Authority) in the Credential Proving protocol is completely independent

from the Credential Issuing protocol: the x'_i and $u'_{i,j}$'s are random, and thus independent from the x_i and $u_{i,j}$'s.

Public Traceability. The subtlety in the new unlinkability assumption (see Definition 5) is that while it hides the fixed value h^u (by applying ElGamal's encryption in $\mathbb{G}_1$), one can still check whether a specific u is included in the credential:

$$e(g^{rt} \cdot h^u, \mathfrak{g}) = e(g^r, \mathfrak{g}^t) \cdot e(h^u, \mathfrak{g}) = e(g^r, \mathfrak{g}^t) \cdot e(h, \mathfrak{g}^u).$$

When we need to trace a user/credential, it suffices to reveal the value $h^* = h^u$ and one can then check whether a credential is associated with this user by testing whether $e(g^{rs} \cdot h^u, \mathfrak{g}) = e(g^r, \mathfrak{g}^s) \cdot e(h^*, \mathfrak{g})$.

With such a public traceability, when a specific data is revealed, it is unavoidable that the user privacy cannot be guaranteed with respect to the Tracing Authority, that knows these tracing data. However, from the evolution of the value H_t among the time periods, $H_t^{u_{i,j}}$ only allows to trace attribute $\mathcal{A}\mathsf{ttr}_j$ for user $\mathcal{X}_i$ during a specific time-period H_t.

5 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 to 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 Extractable Linearly-Homomorphic Signature, this is possible to provide such a proof in an efficient and scalable way.

5.1 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, $\mathcal{B}'_i$ is a randomization of $\mathcal{B}_{\Pi(i)}$. 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.

Initialization

- 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 $\mathsf{Gparam} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, \mathfrak{g}, e, v_{1,1}, v_{1,2}, v_{2,1}, v_{2,2})$;
- System keys: the signing key $\mathsf{SK} = (S_1, S_2, S_3, S_4, S_5, S_6, S_7) \xleftarrow{\$} \mathbb{Z}_p^7$, and the associated verification key $\mathsf{VK} = (g_i = g^{S_i})_i$; and the encryption key: $\mathsf{EK} = h = g^d$ associated to the decryption key $\mathsf{DK} = d \xleftarrow{\$} \mathbb{Z}_p$, and $\ell = g^e$, for $e \xleftarrow{\$} \mathbb{Z}_p$;
- Round parameters: $\mathsf{Rparam} = (g_r = g, h_r = h, \ell_r = \ell, \mathfrak{g}_r = \mathfrak{g}, \mathfrak{s}_r = \mathfrak{s} = \mathfrak{g}^{S_5})$;
- Sender keys: $\mathsf{sk}_i = (u_i, v_i, w_i, y_i; x_i) \xleftarrow{\$} \mathbb{Z}_p^5$ and $\mathsf{vk}_i = (\mathfrak{g}_r; \mathfrak{f}_i = \mathfrak{g}_r^{u_i}, \mathfrak{g}_i = \mathfrak{g}_r^{v_i}, \mathfrak{h}_i = \mathfrak{g}_r^{w_i}, \mathfrak{l}_i = \mathfrak{g}_r^{y_i}; \mathfrak{A}_i = \mathfrak{g}_r^{x_i}, \mathfrak{B}_i = \mathfrak{A}_i^{x_i})$, with a system signature $\Sigma_i = \mathfrak{g}_r^{S_1} \cdot \mathfrak{f}_i^{S_2} \mathfrak{g}_i^{S_3} \mathfrak{h}_i^{S_4} \mathfrak{l}_i^{S_5} \cdot \mathfrak{A}_i^{S_6} \mathfrak{B}_i^{S_7}$

$$e(g, \Sigma_i) = e(g_1, \mathfrak{g}_r) e(g_2, \mathfrak{f}_i) e(g_3, \mathfrak{g}_i) e(g_4, \mathfrak{h}_i) e(g_5, \mathfrak{l}_i) e(g_6, \mathfrak{A}_i) e(g_7, \mathfrak{B}_i) \quad (1)$$

and a proof $\pi_i = (\mathsf{Com}_i, \mathsf{Proof}_i)$ of Square Diffie-Hellman for the tuple $(\mathfrak{g}_r, \mathfrak{A}_i, \mathfrak{B}_i)$: $\mathsf{Com}_i = (c_i = v_{2,1}^{x_i} v_{1,1}^{\mu_i}, d_i = v_{2,2}^{x_i} v_{1,2}^{\mu_i} g^{x_i}) \in \mathbb{G}_1^2$, a commitment of x_i, then $\mathsf{Proof}_i = (\Theta_i = \mathfrak{g}_r^{\mu_i}, \Psi_i = \mathfrak{A}_i^{\mu_i}) \in \mathbb{G}_2^2$ satisfies:

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

$$e(c_i, \mathfrak{A}_i) = e(v_{2,1}, \mathfrak{B}_i) \cdot e(v_{1,1}, \Psi_i) \qquad e(d_i, \mathfrak{A}_i) = e(v_{2,2} g, \mathfrak{B}_i) \cdot e(v_{1,2}, \Psi_i) \quad (3)$$

Ciphertexts

Ciphertexts: $C_i = (g, a_i = g^{r_i}, b_i = h^{r_i} M_i, p_i = \ell^{r_i})$, $C_0 = (1, g_r, h_r, \ell_r)$, with two signatures $\sigma_i = g^{u_i} a_i^{v_i} b_i^{w_i} p_i^{y_i}$ of C_i and $\tau_i = g_r^{v_i} h_r^{w_i} \ell_r^{y_i}$ of C_0:

$$e(\sigma_i, \mathfrak{g}_r) = e(g, \mathfrak{f}_i) e(a_i, \mathfrak{g}_i) e(b_i, \mathfrak{h}_i) e(p_i, \mathfrak{l}_i) \quad e(\tau_i, \mathfrak{g}_r) = e(g_r, \mathfrak{g}_i) e(h_r, \mathfrak{h}_i) e(\ell_r, \mathfrak{l}_i). \quad (4)$$

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 $\mathbb{G}_2$:

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

Mixing

After having verified all ballots (valid with distinct $\mathfrak{A}_i$ different of 1), the mix-server chooses random scalars $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_p$, and publishes new round parameters $\mathsf{Rparam}' = (g_r' = g_r^\beta, h_r' = h_r^\beta, \ell_r' = \ell_r^\beta, \mathfrak{g}_r' = \mathfrak{g}_r^\alpha, \mathfrak{s}_r' = \mathfrak{s}_r^\alpha)$: for each ballot, it chooses $\gamma_i, \delta_i, z_i \xleftarrow{\$} \mathbb{Z}_p$, and sets

$$\mathsf{vk}_i' = (\mathfrak{g}_r'; \mathfrak{f}_i' = \mathfrak{f}_i^\alpha, \mathfrak{g}_i' = \mathfrak{g}_i^\alpha, \mathfrak{h}_i' = \mathfrak{h}_i^\alpha, \mathfrak{l}_i' = \mathfrak{l}_i^\alpha \mathfrak{g}_r'^{z_i}; \mathfrak{A}_i' = \mathfrak{A}_i^\alpha, \mathfrak{B}_i' = \mathfrak{B}_i^\alpha)$$

$$C_i' = (g, a_i' = a_i g_r^{\gamma_i}, b_i' = b_i h_r^{\gamma_i}, p_i' = p_i \ell_r^{\gamma_i}) \qquad C_0' = (1, g_r', h_r', \ell_r')$$

$$\Sigma_i' = \Sigma_i^\alpha \cdot \mathfrak{s}_r'^{z_i} \qquad \sigma_i' = \sigma_i \cdot \tau_i^{\gamma_i} \cdot p_i'^{z_i} \qquad \tau_i' = \tau_i^\beta \cdot \ell_r'^{z_i}$$

$$c_i' = c_i \cdot v_{1,1}^{\delta_i} \qquad d_i' = d_i \cdot v_{1,2}^{\delta_i} \qquad \Theta_i' = (\Theta_i \cdot \mathfrak{g}_r^{\delta_i})^\alpha \qquad \Psi_i' = (\Psi_i \cdot \mathfrak{A}_i^{\delta_i})^\alpha$$

The new tuples are then outputted in random order. The global proof just consists in a proof of Diffie-Hellman tuple for $(\mathfrak{g}_r, \mathfrak{g}_r', \prod_i \mathfrak{A}_i, \prod_i \mathfrak{A}_i')$. The verifier will check this proof, 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 the equations (1), (2), (3), and (4), are satisfied on individual output tuples.

Fig. 2. Shuffling of ElGamal Ciphertexts

5.2 Our Mixing Protocol

The complete and precise description is presented in Figure 2, but we provide here an intuition of the construction: we 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 $z_i = 0$ (see Figure 2) to have a perfect linear randomization (such linearity can lead to linkability, hence the alteration z_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, $z_i = 0$) 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 $\mathsf{Rparam} = (g_r, h_r, \ell_r, \mathfrak{g}_r, \mathfrak{s}_r)$ initially set to $(g, h, \ell, \mathfrak{g}, \mathfrak{s})$, where $\mathfrak{s}$ is a signature of $\mathfrak{g}$ under g_5 as $e(g, \mathfrak{s}) = e(g_5, \mathfrak{g})$. Note that this signature $\mathfrak{s}$ will just be useful when $z_i \neq 0$, 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 $(g, g^{r_i}, h^{r_i} M_i, \ell^{r_i})$ of M_i and the ciphertext $(1, g_r, h_r, \ell_r)$ 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 last component that excludes linear attacks (when $z_i \neq 0$). Indeed, since only linear combinations are possible, one can just sign some pair $C'_i = (g^{t_i}, a'_i = g^{t_i(r_i+\gamma_i)}, b'_i = h^{t_i(r_i+\gamma_i)} M_i^{t_i}, p'_i = \ell^{t_i(r_i+\gamma_i)})$, 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 $z_i \neq 0$, 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 new encryption of 1, for a next randomization: $C'_0 = (1, g'_r = g_r^{\beta}, h'_r = h_r^{\beta}, \ell'_r = \ell_r^{\beta})$, for which $\tau'_i = \tau_i^{\beta}$ is a signature (again, when $z_i \neq 0$, thanks to $\mathfrak{s}$, one can alter vk'_i and adapt τ'_i):

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

   Thanks to the linear restriction with τ_i and τ'_i, since the first components of both C_0 and C'_0 are imposed to be 1, this guarantees the multi-Diffie-Hellman tuples $(g_r, h_r, \ell_r, g'_r, h'_r, \ell'_r)$. If C_0 was a ciphertext of 1, this is also true for C'_0.
2. one needs non-miscibility of the verifications keys to guarantee $\mathsf{vk}'_i = \mathsf{vk}_i^{\alpha_i}$ (when $z_i = 0$). To this aim, we will consider $\mathsf{vk}_i = (\mathfrak{g}_r; \mathfrak{f}_i, \mathfrak{g}_i, \mathfrak{h}_i, \mathfrak{l}_i; \mathfrak{g}_r^{x_i}, \mathfrak{g}_r^{x_i^2})$

for signing quadruples (the above ciphertexts C_0 and C_i) with the four components $(\mathfrak{f}_i, \mathfrak{g}_i, \mathfrak{h}_i, \mathfrak{l}_i)$, while the three other components $(\mathfrak{g}_r, \mathfrak{g}_r^{x_i}, \mathfrak{g}_r^{x_i^2})$ guarantee non-miscibility, for unknown and random $x_i \xleftarrow{\$} \mathbb{Z}_p$, as shown in Theorem 9. C_i is converted into C'_i signed under the new verification key $\mathsf{vk}'_i = (\mathfrak{g}_r^\alpha; \mathfrak{f}_i^\alpha, \mathfrak{g}_i^\alpha, \mathfrak{h}_i^\alpha, \mathfrak{l}_i^\alpha; \mathfrak{g}_r^{\alpha x_i}, \mathfrak{g}_r^{\alpha x_i^2})$, with a valid signature $\Sigma'_i = \Sigma_i^\alpha$ (still when $z_i = 0$), under VK:

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

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 2, one has

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

In the initial proof Π_i, Com_i is a Groth-Sahai [GS08] commitment of x_i and Proof_i is the proof that both $\mathfrak{A}_i = \mathfrak{g}_r^{x_i}$ and $\mathfrak{B}_i = \mathfrak{A}_i^{x_i}$. Then, one just randomizse the commitment Com_i into Com'_i of the same value x_i, which allows the update of Proof_i into Proof'_i, as detailed in figure 2.

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 x_i's are in both products: indeed, as all individual ballots are signed, they must contain their original x_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 $(\mathfrak{g}'_r = \mathfrak{g}_r^\alpha, \mathfrak{s}'_r = \mathfrak{s}_r^\alpha, \mathfrak{f}'_i = \mathfrak{f}_i^\alpha, \mathfrak{g}'_i = \mathfrak{g}_i^\alpha, \mathfrak{h}'_i = \mathfrak{h}_i^\alpha, \mathfrak{A}'_i = \mathfrak{A}_i^\alpha, \mathfrak{B}'_i = \mathfrak{B}_i^\alpha)$, as well as $(g'_r = g_r^\beta, h'_r = h_r^\beta, \ell'_r = \ell_r^\beta)$, the new tuples $(a'_i, b'_i, p'_i, \sigma'_i, \tau'_i)$, $(\mathfrak{g}'_r; \mathfrak{f}'_i, \mathfrak{g}'_i, \mathfrak{h}'_i, \mathfrak{l}'_i; \mathfrak{A}'_i, \mathfrak{B}'_i)$, Σ'_i, $(c'_i, d'_i, \Theta'_i, \Psi'_i)$ satisfy the initial relations (1), (2), (3), (4), with respect to the new round parameters $\mathsf{Rparam}' = (g'_r, h'_r, \ell'_r, \mathfrak{g}'_r, \mathfrak{s}'_r)$. One can thus re-iterate the mixing process.

5.3 Efficiency

As shown in Figure 2, 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 9 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: 3 exponentiations in $\mathbb{G}_1$ and 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 3) 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.

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 $(\mathfrak{g}_r^{(0)}, \mathfrak{g}_r^{(k)}, \mathfrak{A}^{(0)}, \mathfrak{A}^{(k)} = \prod \mathfrak{A}_i^{(k)})$,

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

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.

Fig. 3. Efficient Mix-Net with Multiple 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 3, 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 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.

6 Security Analysis

Let us now formally prove the security properties, which should be:

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

6.1 Non-Miscibility of the Verification Keys

The first step is the guarantee that every output verification key vk'_i corresponds to a legitimate verification vk_i. From the legitimate verification keys $(\mathsf{vk}_i = (\mathfrak{g}_r; \mathfrak{f}_i, \mathfrak{g}_i, \mathfrak{h}_i, \mathfrak{l}_i; \mathfrak{A}_i, \mathfrak{B}_i), \Sigma_i)_i$ and $(\mathsf{rk} = (1, 1, 1, 1, \mathfrak{g}_r, 1, 1), \mathfrak{s})$, signed under VK, such that each $(\mathfrak{g}_r, \mathfrak{A}_i, \mathfrak{B}_i)$ is a Square Diffie-Hellman tuple, any mix-server must output a new verification key vk'_i with a new signature Σ'_i (because of equation (1) on the output tuples). As shown in Theorem 6, vk'_i must be a linear combination of the $(\mathsf{vk}_j)_j$ and rk, with coefficients $(\alpha_{i,j})_j$ and z_i. Since vk'_i also contains a Square Diffie-Hellman tuple (because of equations (2) and (3) on the output tuples), Theorem 9 guarantees that at most one coefficient α_{i,j_i} is non-zero: if they are all zeros, then $\mathsf{vk}'_i = (1, 1, 1, 1, *, 1, 1)$, which will be rejected as $\mathfrak{A}'_i = 1$; then for each i, $\mathsf{vk}'_i = (\mathsf{vk}_{j_i} \times \mathsf{rk}^{z_i})^{\alpha_{i,j_i}}$ for some index j_i. Since we impose the first component of vk'_i, denoted $\mathfrak{g}'_r$, to be the same for everybody, then for every i, $\alpha_{j_i} = \alpha$ such that $\mathfrak{g}'_r = \mathfrak{g}_r^{\alpha}$. We can then notice that the fifth element $\mathfrak{l}'_i$ of vk'_i is of the form $\mathfrak{l}'_i = (\mathfrak{l}_i \times \mathfrak{g}_r^{z_i})^{\alpha}$, for some integer z_i, but all the other ones are α-power of the corresponding element in vk_{j_i}. This proves the following proposition:

Proposition 11 (Legitimate Output). *For any output ballot with verification key vk'_i there exists a related legitimate verification key vk_j such that $\mathsf{vk}'_i = (\mathsf{vk}_j \times \mathsf{rk}^{z_i})^{\alpha}$, for some scalar z_i, and α such that $\mathfrak{g}'_r = \mathfrak{g}_r^{\alpha}$*

We stress that for the proposition to hold, we need the x_i's to be unknown to the adversary. But 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. Note however that no voter can have two ballots, because of the collision checks on the $\mathfrak{A}'_i$'s, if there was no collision on the $\mathfrak{A}_i$'s.

6.2 Permutation of the Verification Keys

Hence, the second step consists in proving that any output verification key corresponds to an input verification key. 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 Theorem 6, as used above, 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 x_{j_i} that was not in the initial ballot-box.

We will again exploit the fact that nobody knows the x_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 x_i with randomness μ_i as $(c_i, d_i) = (v_{1,1}^{\mu_i + \rho x_i}, v_{1,2}^{\mu_i + \rho x_i})$, the commitment of 0 with randomness $\nu_i = \mu_i + \rho x_i$. Hence, $\Theta_i = \mathfrak{g}_r^{\mu_i} = \mathfrak{g}_r^{\nu_i - \rho x_i} = \mathfrak{A}_i^{-\rho} \cdot \mathfrak{g}_r^{\nu_i}$ and similarly $\Psi_i = \mathfrak{A}_r^{\mu_i} = \mathfrak{A}_r^{\nu_i - \rho x_i} = \mathfrak{B}_i^{-\rho} \cdot \mathfrak{A}_r^{\nu_i}$.
- This new setup allows to simulate the proofs without knowing x_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 $\mathfrak{A}_i \leftarrow \mathfrak{g}^{x_i} \mathfrak{h}^{y_i}$, and $\mathfrak{B}_i \leftarrow \mathfrak{g}^{z_i}$ for random scalars $x_i, y_i, z_i \xleftarrow{\$} \mathbb{Z}_p$, with an independent group element $\mathfrak{h} \xleftarrow{\$} \mathbb{G}_2$.

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

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

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

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

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 $\mathfrak{g}$. This proves the following proposition:

Proposition 12 (Permutation of Keys). *Given a set of legitimate verification keys vk_i's with random and unknown x_i's, from any input ballot-box (with verification keys vk_j, for $j \in \mathcal{J}$) and transformed ballot-box (with verification key vk'_i, for $i \in \mathcal{I}$) such that all the $\mathfrak{A}_j$'s are distinct and different of 1, all the $\mathfrak{A}'_i$'s are distinct and different of 1, $|\mathcal{I}| = |\mathcal{J}|$, and $(\mathfrak{g}_r, \mathfrak{g}'_r, \prod_j \mathfrak{A}_j, \prod_i \mathfrak{A}'_i)$ is a Diffie-Hellman, then there exists α and a bijection Π from $\mathcal{I}$ into $\mathcal{J}$ such that for any i, $\mathsf{vk}'_i = (\mathsf{vk}_{\Pi(i)} \times \mathsf{rk}^{z_i})^\alpha$ for some scalar z_i, unless one can break the* DL *assumption in $\mathbb{G}_2$, the* DDH *assumption in $\mathbb{G}_1$ and the* DSDH *assumption in $\mathbb{G}_2$ (already implied by the* DL *assumption in $\mathbb{G}_2$).*

We stress that for this property to hold, the scalars x_i's must be random and unknown to the mix-server and the senders.

6.3 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 vk'_i: there exists a permutation Π of $\{1,\ldots,n\}$ such that $\mathsf{vk}'_i = (\mathsf{vk}_{\Pi(i)} \times \mathsf{rk}^{z_i})^\alpha$ for some scalar z_i.

From the equation (4) on the output tuples, C'_i is signed under vk'_i in σ'_i, for every i: $e(\sigma'_i, \mathfrak{g}'_r) = e(g, \mathfrak{f}^\alpha_{\Pi(i)}) \cdot e(a'_i, \mathfrak{g}^\alpha_{\Pi(i)}) \cdot e(b'_i, \mathfrak{h}^\alpha_{\Pi(i)}) \cdot e(p'_i, \mathfrak{l}^\alpha_{\Pi(i)} \mathfrak{g}_r^{\alpha z_i})$, and since the same α appears in $\mathfrak{g}'_r = \mathfrak{g}_r^\alpha$, then for every i, $e(\sigma'_i, \mathfrak{g}_r) = e(g, \mathfrak{f}_{\Pi(i)}) \cdot e(a'_i, \mathfrak{g}_{\Pi(i)} \cdot e(b'_i, \mathfrak{h}_{\Pi(i)}) \cdot e(p'_i, \mathfrak{l}_{\Pi(i)}) \cdot e(p'^{z_i}_i, \mathfrak{g}_r)$. Then $\sigma'_i / p'^{z_i}_i$ is a signature of $C'_i = (g, a'_i, b'_i, p'_i)$ under $\mathsf{vk}_{\Pi(i)}$: from Theorem 6, C'_i is necessarily a linear combination of the already signed vectors under $\mathsf{vk}_{\Pi(i)}$, which are $C_{\Pi(i)}$ and C_0, with coefficients u, v: $a'_i = a^u_{\Pi(i)} g_r^v$, $b'_i = b^u_{\Pi(i)} h_r^v$, and $g = g^u 1^v$. Hence, $u = 1$, which means that C'_i is a randomization of $C_{\Pi(i)}$. Similarly, C'_0 is signed under vk'_i in τ'_i, for every i: $e(\tau'_i, \mathfrak{g}'_r) = e(g'_r, \mathfrak{g}^\alpha_{\Pi(i)}) \cdot e(h'_r, \mathfrak{h}^\alpha_{\Pi(i)}) \cdot e(\ell'_r, \mathfrak{l}^\alpha_{\Pi(i)} \mathfrak{g}_r^{\alpha z_i})$, which means that $e(\tau'_i, \mathfrak{g}_r) = e(g'_r, \mathfrak{g}_{\Pi(i)}) \cdot e(h'_r, \mathfrak{h}_{\Pi(i)}) \cdot e(\ell'_r, \mathfrak{l}_{\Pi(i)}) \cdot e(\ell'^{z_i}_r, \mathfrak{g}_r)$. Then $\tau'_i / \ell'^{z_i}_r$ is a signature of $C'_0 = (1, g'_r, h'_r, \ell'_r)$ under $\mathsf{vk}_{\Pi(i)}$: from Theorem 6, C'_0 is necessarily a linear combination of the already signed vectors under $\mathsf{vk}_{\Pi(i)}$, which are $C_{\Pi(i)}$ and C_0, with coefficients u', v': $g'_r = a^{u'}_{\Pi(i)} g_r^{v'}$, $h'_r = b^{u'}_{\Pi(i)} h_r^{v'}$, and $1 = g^{u'} 1^{v'}$. Hence, $u' = 0$, which means that C'_0 is a randomization of C_0: the existence of β is implicitly proven:

Proposition 13 (Permutation of Ciphertexts). *Given a set of legitimate verification keys vk_i's, from any input ballot-box (with verification keys vk_j and ciphertexts C_j, for $j \in \mathcal{J}$) and transformed ballot-box (with verification key vk'_i and ciphertexts C_i, for $i \in \mathcal{I}$) such that all the $\mathfrak{A}_j$'s are distinct and different of 1, all the $\mathfrak{A}'_i$'s are distinct and different of 1, $|\mathcal{I}| = |\mathcal{J}|$, and $(\mathfrak{g}_r, \mathfrak{g}'_r, \prod_j \mathfrak{A}_j, \prod_i \mathfrak{A}'_i)$ is a Diffie-Hellman, then there exists a bijection Π from $\mathcal{I}$ into $\mathcal{J}$ such that for any i, C'_i is a randomization of $C'_{\Pi(i)}$, unless one can break the* DL *assumption*

in $\mathbb{G}_2$*, the* DDH *assumption in* $\mathbb{G}_1$ *and the* DSDH *assumption in* $\mathbb{G}_2$ *(already implied by the* DL *assumption in* $\mathbb{G}_2$*).*

6.4 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 multiple tuples of several users, if the simulator randomizes the tuple of user $k \in \{i, j\}$, no adversary can guess whether $k = i$ or $k = j$. From the proof, we will then be able to discuss unlinkability with respect to insider verifiers (also required for receipt-freeness in the case of electronic voting). Our proof is in the same vein as for the unlinkability of anonymous credentials (see Section 4, but we will provide it in full details. We start from the initial real game that generates the view of the adversary, for the initial ballots and the randomized ballots (for the same user) and conclude with a game where the randomized ballots are completely anonymous, and thus unlinkable to the initial ones:

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 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 $\mathsf{Rparam} = (g_r = g, h_r = h, \ell_r = \ell, \mathfrak{g}_r = \mathfrak{g}, \mathfrak{s}_r = \mathfrak{g}_r^{S_5})$, as well as $u_i, v_i, w_i, x_i, y_i, \mu_i, r_i \xleftarrow{\$} \mathbb{Z}_p$ to generate $\mathsf{vk}_i = (\mathfrak{g}_r; \mathfrak{f}_i = \mathfrak{g}_r^{u_i}, \mathfrak{g}_i = \mathfrak{g}_r^{v_i}, \mathfrak{h}_i = \mathfrak{g}_r^{w_i}, \mathfrak{l}_i = \mathfrak{g}_r^{y_i}; \mathfrak{A}_i = \mathfrak{g}_r^{x_i}, \mathfrak{B}_i = \mathfrak{A}_i^{x_i})$, Σ_i, $\pi_i = (\mathsf{Com}_i, \mathsf{Proof}_i)$, $C_i = (g, a_i = g_r^{r_i}, b_i = h_r^{r_i} M_i, p_i = \ell_r^{r_i})$ and $C_0 = (1, g_r, h_r, \ell_r)$, with σ_i and τ_i. Indeed, knowing the signing keys allows to honestly generate the deterministic signatures Σ_i, σ_i, and τ_i. This is done for several users (at least two, i and j), then one chooses $k \in \{i, j\}$. Actually, we will continue with user i, and show the randomization will be independent from any of this user's parameters.
By randomly choosing α, β and γ_i, δ_i, z_i in $\mathbb{Z}_p$, the simulator also generates, as the mix-server would do, $\mathsf{Rparam}' = (g'_r = g_r^\beta, h'_r = h_r^\beta, \ell'_r = \ell_r^\beta, \mathfrak{g}'_r = \mathfrak{g}_r^\alpha, \mathfrak{s}'_r = \mathfrak{s}_r^\alpha)$ with a proof of knowledge of α such that $\mathfrak{g}'_r = \mathfrak{g}_r^\alpha$, and

$$\mathsf{vk}'_i = (\mathfrak{g}'_r; \mathfrak{f}'_i = \mathfrak{f}_i^\alpha, \mathfrak{g}'_i = \mathfrak{g}_i^\alpha, \mathfrak{h}'_i = \mathfrak{h}_i^\alpha, \mathfrak{l}'_i = \mathfrak{l}_i^\alpha \mathfrak{g}_r'^{z_i}; \mathfrak{A}'_i = \mathfrak{A}_i^\alpha, \mathfrak{B}'_i = \mathfrak{B}_i^\alpha)$$
$$C'_i = (g, a'_i = a_i g_r^{\gamma_i}, b'_i = b_i h_r^{\gamma_i}, p'_i = p_i \ell_r^{\gamma_i}) \qquad C'_0 = (1, g'_r, h'_r, \ell'_r)$$

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

Game $\mathbf{G}_1$: Our first step is to 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.

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, 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 x_i with randomness μ_i as $(c_i, d_i) = (v_{1,1}^{\mu_i+\rho x_i}, v_{1,2}^{\mu_i+\rho x_i})$, the commitment of 0 with randomness $\nu_i = \mu_i + \rho x_i$. Hence, $\Theta_i = \mathfrak{g}_r^{\mu_i} = \mathfrak{g}_r^{\nu_i - \rho x_i} = \mathfrak{A}_i^{-\rho} \cdot \mathfrak{g}_r^{\nu_i}$ and similarly $\Psi_i = \mathfrak{A}_i^{\mu_i} = \mathfrak{A}_i^{\nu_i - \rho x_i} = \mathfrak{B}_i^{-\rho} \cdot \mathfrak{A}_i^{\nu_i}$. From the modification applied with δ_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 $\nu'_i = \nu_i + \delta_i$.

Game $\mathbf{G_3}$: Since we do not need x_i anymore for the proofs, the simulator receives a random Square Diffie-Hellman tuple $(\mathfrak{g}_r, \mathfrak{A}_i, \mathfrak{B}_i)$, and everything else can be simulated as above, without x_i but with random ν_i, ν'_i and using ρ. In particular, the public keys can be generated as

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

from the known scalars u_i, v_i, w_i, y_i, z_i.

Game $\mathbf{G_4}$: The simulator now receives a random tuple $(\mathfrak{g}_r, \mathfrak{A}_i, \mathfrak{B}_i)$ and runs as before. This game is indistinguishable from the previous one thanks to the Square Diffie-Hellman assumption.

Game $\mathbf{G_5}$: We now do the simulation with

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

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

Game $\mathbf{G_6}$: Since the decryption key d is never used, we are just given $\mathsf{EK} = h = h_r$, and can replace C_i by an encryption of 1. Under the DDH assumption in $\mathbb{G}_1$, this is indistinguishable.

Game $\mathbf{G_7}$: Since all the ciphertexts encrypt 1, we can simulate them, with $(g_r, h_r, \ell_r) = (g, g^d, g^e)$ and $(g'_r, h'_r, \ell'_r) = (g_r, h_r, \ell_r)^{\beta}$ for C_0 and C'_0, and $(a_i, b_i, p_i) = (g_r, h_r, \ell_r)^{r_i}$ and $(a'_i, b'_i, p'_i) = (g'_r, h'_r, \ell'_r)^{r'_i}$ for C_i and C'_i, for known random scalars $d, e, r_i, \gamma_i, \beta \xleftarrow{\$} \mathbb{Z}_p$, and $r'_i = r_i + \gamma_i$. The proofs and signatures are still simulated as above using the known scalars u_i, v_i, w_i, y_i, $x_i, \bar{x}_i, z_i, \alpha \xleftarrow{\$} \mathbb{Z}_p$.

Let us just make some remarks about the generated signatures: they satisfy equations (4) on both the input and output ciphertexts. Thus,

$$e(\sigma_i, \mathfrak{g}_r) = e(g, \mathfrak{f}_i) \cdot e(a_i, \mathfrak{g}_i) \cdot e(b_i, \mathfrak{h}_i) \cdot e(p_i, \mathfrak{l}_i) = e(g, \mathfrak{f}_i(\mathfrak{g}_i\mathfrak{h}_i^d\mathfrak{l}_i^e)^{r_i})$$
$$e(\tau_i, \mathfrak{g}_r) = e(g_r, \mathfrak{g}_i) \cdot e(h_r, \mathfrak{h}_i) \cdot e(\ell_r, \mathfrak{l}_i) = e(g_r, \mathfrak{g}_i\mathfrak{h}_i^d\mathfrak{l}_i^e)$$
$$e(\sigma'_i, \mathfrak{g}'_r) = e(g, \mathfrak{f}'_i) \cdot e(a'_i, \mathfrak{g}'_i) \cdot e(b'_i, \mathfrak{h}'_i) \cdot e(p'_i, \mathfrak{l}'_i) = e(g, \mathfrak{f}'_i(\mathfrak{g}'_i\mathfrak{h}_i'^d\mathfrak{l}_i'^e)^{\beta r'_i})$$
$$e(\tau'_i, \mathfrak{g}'_r) = e(g'_r, \mathfrak{g}'_i) \cdot e(h'_r, \mathfrak{h}'_i) \cdot e(\ell'_r, \mathfrak{l}'_i) = e(g'_r, \mathfrak{g}'_i\mathfrak{h}_i'^d\mathfrak{l}_i'^e)$$

If we formally denote $\sigma_i = g^{s_i}$, $\tau_i = g_r^{t_i}$, $\sigma'_i = g^{s'_i}$, $\tau'_i = {g'_r}^{t'_i}$, then

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

and from

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

we also have

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

Game $\mathbf{G}_8$: Let us randomly choose scalars $u_i, r_i, r'_i, t_i, t'_i$ and α, β, from $(g_r, \mathfrak{g}_r)$, we can set $g'_r \leftarrow g_r^{\beta}$ and $\mathfrak{g}'_r \leftarrow \mathfrak{g}_r^{\alpha}$, $a_i \leftarrow g_r^{r_i}$, $\sigma_i \leftarrow a_i^{t_i} g^{u_i}$, $\mathfrak{f}_i \leftarrow \mathfrak{g}_r^{u_i}$, as well as $a'_i \leftarrow {g'_r}^{r'_i}$, $\sigma'_i \leftarrow {a'_i}^{t'_i} g^{u_i}$, $\mathfrak{f}'_i \leftarrow {\mathfrak{g}'_r}^{u_i}$.
Then, one additionally chooses $v_i, w_i \xleftarrow{\$} \mathbb{Z}_p$ and sets

$$\begin{aligned}\mathfrak{g}_i &\leftarrow \mathfrak{g}_r^{v_i} \quad \mathfrak{h}_i \leftarrow \mathfrak{g}_r^{w_i} & \mathfrak{g}'_i &\leftarrow {\mathfrak{g}'_r}^{v_i} \quad \mathfrak{h}'_i \leftarrow {\mathfrak{g}'_r}^{w_i}\\ \mathfrak{l}_i &\leftarrow (\mathfrak{g}_r^{t_i}/(\mathfrak{g}_i\mathfrak{h}_i^d))^{1/e} & \mathfrak{l}'_i &\leftarrow ({\mathfrak{g}'_r}^{t'_i}/(\mathfrak{g}'_i{\mathfrak{h}'_i}^d))^{1/e}\\ C_0 &\leftarrow (1, g_r, g_r^d, g_r^e) & C'_0 &\leftarrow (1, g'_r, {g'_r}^d, {g'_r}^e)\\ C_i &\leftarrow (g, a_i, a_i^d, a_i^e) & C'_i &\leftarrow (g, a'_i, {a'_i}^d, {a'_i}^e)\end{aligned}$$

By construction

$$\begin{aligned}\mathfrak{g}_r^{t_i} &= \mathfrak{g}_i\mathfrak{h}_i^d\mathfrak{l}_i^e & {\mathfrak{g}'_r}^{t'_i} &= \mathfrak{g}'_i{\mathfrak{h}'_i}^d{\mathfrak{l}'_i}^e\\ \sigma_i &= a_i^{t_i} g^{u_i} = g^{t_i r_i} \times g^{u_i} & \sigma'_i &= {a'_i}^{t'_i} g^{u'_i} = {g'_r}^{t'_i r'_i} \times g^{u_i}\end{aligned}$$

So σ_i and σ'_i are valid signatures of C_i and C'_i, and $\tau_i \leftarrow g_r^{t_i}$ and $\tau'_i \leftarrow {g'_r}^{t'_i}$ are valid signatures of C_0 and C'_0. Eventually, with $u'_i = u_i$, the public keys $(\mathfrak{g}_r, \mathfrak{f}_i, \mathfrak{g}_i, \mathfrak{h}_i)$, $(\mathfrak{g}'_r, \mathfrak{f}'_i, \mathfrak{g}'_i, \mathfrak{h}'_i)$ are correctly related for the secret keys (u_i, v_i, w_i). About $\mathfrak{l}_i = (\mathfrak{g}_r^{t_i}/(\mathfrak{g}_i\mathfrak{h}_i^d))^{1/e} = \mathfrak{g}_r^{(t_i - v_i - dw_i)/e}$. Hence we have $y_i = (t_i - v_i - dw_i)/e$, while $\mathfrak{l}'_i = ({\mathfrak{g}'_r}^{t'_i}/(\mathfrak{g}'_i{\mathfrak{h}'_i}^d))^{1/e} = {\mathfrak{g}'_r}^{(t'_i - v_i - dw_i)/e}$, hence $y'_i = (t'_i - v_i - dw_i)/e = (t'_i - t_i)/e + y_i$, which means that $z_i = (t'_i - t_i)/e$.
With random $x_i, \bar{x}_i$, using the secret key $\mathsf{SK} = (S_i)_i$, we can complete and sign vk_i and vk'_i. We also simulate the proofs for $(\mathfrak{A}_i, \mathfrak{B}_i, \mathfrak{A}'_i, \mathfrak{B}'_i)$. As explained above, this perfectly simulate the view of the adversary in the previous game.

Game $\mathbf{G}_9$: Let us be given two credentials of u_i, $\mathsf{Cred}(u_i, g; g_r, \mathfrak{g}_r, r_i, t_i)$ and $\mathsf{Cred}(u_i, g; g'_r, \mathfrak{g}'_r, r'_i, t'_i)$, 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 $\mathcal{D}_{g,g_r,\mathfrak{g}_r}(u_i, u_i)$. We can thus continue the simulation as above, in a perfectly indistinguishable way.

Game $\mathbf{G}_{10}$: Let us be given two credentials of u_i and u'_i, $\mathsf{Cred}(u_i, g; g_r, \mathfrak{g}_r, r_i, t_i)$ and $\mathsf{Cred}(u'_i, g; g'_r, \mathfrak{g}'_r, r'_i, t'_i)$, for a random $u_i, u'_i \xleftarrow{\$} \mathbb{Z}_p$. Inputs follow the distribution $\mathcal{D}_{g,g_r,\mathfrak{g}_r}(u_i, u'_i)$. And we do as above. Under the Unlinkability Assumption (see Definition 5) the view is computationally indistinguishable.

Game $\mathbf{G}_{11}$: Now, we receive a Multi Diffie-Hellman tuple $(\mathfrak{g}_r, \mathfrak{g}_i, \mathfrak{h}_i, \mathfrak{A}_i, \mathfrak{B}_i, \mathfrak{g}'_r, \mathfrak{g}'_i, \mathfrak{h}'_i, \mathfrak{A}'_i, \mathfrak{B}'_i) \xleftarrow{\$} \mathcal{D}^{10}_{\mathsf{mdh}}(\mathfrak{g}_r)$. So we know all the scalars, except $v_i, w_i, x_i, \bar{x}_i$ 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 $r_i, r'_i, \beta \xleftarrow{\$} \mathbb{Z}_p$.

Game $\mathbf{G}_{12}$: We now receive $(\mathfrak{g}_r, \mathfrak{g}_i, \mathfrak{h}_i, \mathfrak{A}_i, \mathfrak{B}_i, \mathfrak{g}'_r, \mathfrak{g}'_i, \mathfrak{h}'_i, \mathfrak{A}'_i, \mathfrak{B}'_i) \xleftarrow{\$} \mathcal{D}^{10}_{\$}(\mathfrak{g}_r)$. We do the simulation as above. The view of the adversary is indistinguishable under the DDH assumption in $\mathbb{G}_2$.

In this final game, vk'_i is a random key, for an extended secrete key $\mathsf{sk}'_i = (u'_i, v'_i, w'_i, y'_i; x'_i, \bar{x}'_i) \xleftarrow{\$} \mathbb{Z}_p^6$, independently of the extended secrete key $\mathsf{sk}_i = (u_i, v_i, w_i, y_i; x_i, \bar{x}_i)$. The ciphertext C'_0, C'_i are all random encryption of 1, independently of the initial ciphertexts, commitments all commit to 0, and signatures and proofs are deterministic, hence there is no way to distinguish a randomization from user i and a randomization from user j:

Proposition 14 (Unlinkability of the Mixing). *Given any input ballot-box and transformed ballot-box (following the protocol from Figure 2), for any adversary that chooses two input ballots $\mathcal{B}_0$ and $\mathcal{B}_1$ and is given back the transformation $\mathcal{B}'$ of $\mathcal{B}_b$ for a random bit $b \xleftarrow{\$} \{0,1\}$, the advantage in guessing b is negligible.*

We stress that for this property to hold, the adversary should not know the two secret keys sk_{i_0} and sk_{i_1}, where $\mathsf{sk}_i = (u_i, v_i, w_i, y_i; x_i)$. Even if we already assumed x_i unknown to anybody, we did not say anything about (u_i, v_i, w_i, y_i). The indistinguishability is for outsider adversaries only.

7 Application to Electronic Voting

7.1 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 x_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, w_i, y_i) but should jointly generate $(\mathfrak{A}_i, \mathfrak{B}_i)$ together with the proof π_i such that nobody knows x_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 $x_{i,1} \xleftarrow{\$} \mathbb{Z}_p$ and computes

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

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

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

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

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

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

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

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

Then, the signer publishes

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

together with the proof

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

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

The signature Σ_i can only be used by the voter who knows $\mathsf{sk}_i = (u_i, v_i, w_i, y_i)$ for the verification key $\mathsf{vk}_i = (\mathfrak{f}_i, \mathfrak{g}_i, \mathfrak{h}_i, \mathfrak{l}_i; \mathfrak{A}_i, \mathfrak{B}_i)$. But the proof of knowledge of $x_{i,2}$ guarantees that x_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 $x_{i,2}$ such that $\mathfrak{A}_{i,2} = \mathfrak{A}_i/\mathfrak{A}_{i,1} = \mathfrak{g}_r^{x_{i,2}}$;
– we now receive a random tuple $(\mathfrak{g}_r, \mathfrak{A}_i, \mathfrak{B}_i)$, for unknown x_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.

7.2 Receipt-Freeness

For electronic voting, receipt-freeness is an important security property, but hard to achieve: 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, but just $\mathfrak{A}_i$ for each voter, the voter cannot exploit his encryption random coins. But he can use (u_i, v_i, w_i, y_i) to prove which randomized ballot corresponds to his vote. After decryption, this will prove his vote. The voter should not know (u_i, v_i, w_i, y_i) either (not just x_i as above). This can be easily done by the voter choosing $(u_{i,1}, v_{i,1}, w_{i,1}, y_{i,1}) \xleftarrow{\$} \mathbb{Z}_p^4$ to compute

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

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

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

The latter can also compute

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

and generate the signature Σ_i, together with zero-knowledge proofs of knowledge of the scalars $u_{i,2}, v_{i,2}, w_{i,2}, y_{i,2}$. 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 and τ_i. However, the privacy of the vote is still guaranteed as it is encrypted under EK.

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.

CL04. Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 56–72. Springer, Heidelberg, August 2004.

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.

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.

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.

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 *ACM CCS 01*, 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.