Linearly-Homomorphic Signatures for Short Randomizable Proofs of Subset Membership

David Pointcheval

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

Abstract. Electronic voting is one of the most interesting application of modern cryptography, as it involves many innovative tools (such as homomorphic public-key encryption, non-interactive zero-knowledge proofs, and distributed cryptography) to guarantee several *a priori* contradictory security properties: the integrity of the tally and the privacy of the individual votes. While many efficient solutions exist for *honest-but-curious* voters, that follow the official procedure but try to learn more than just the public result, preventing attacks from *malicious* voters is much more complex: when voters may have incentive to send biased ballots, the privacy of the ballots is much harder to satisfy, whereas this is the crucial security property for electronic voting.

We present a new technique to prove that an ElGamal ciphertext contains a message from a specific subset (quasi-adaptive NIZK of subset membership), using linearly-homomorphic signatures. The proofs are both quite efficient to generate, allowing the use of low-power devices to vote, and randomizable, which is important for the strong receipt-freeness property. They are well-suited to prevent vote-selling and replay attacks, which are the main threats against the privacy in electronic voting, with security proofs in the generic group model and the random oracle model.

1 Introduction

With the all-digital society, and more recently with the pandemic and multiple lock-down periods, democracy is moving towards remote electronic voting, *a.k.a.* internet voting. Several solutions have been developed that all encrypt the ballot on the voter-side to guarantee the voter's privacy. Thereafter, two major approaches exist for counting the tally, according to the complexity of the election: either one applies a mixing-network (mixnet), which permutes and randomizes the encrypted ballots, before decryption of all the individual ballots to perform the counting in the clear, as one does with paper-based voting systems when one opens the envelops after having mixed them to remove any link with the voters; or one uses homomorphic encryption that allows to aggregate the encrypted ballots to get the encrypted tally, that is the unique value eventually decrypted. The latter approach is definitely the most appropriate for electronic voting, when the tally just consists in ranking the candidates w.r.t. their number of votes, as it allows a fast publication of the results, while mixing the ballots in a verifiable way is time-consuming for large scale elections. We will thus target this approach in the following, with the ElGamal encryption scheme.

Replay Attacks and Vote-Selling. However, privacy is more complex than it appears: simple encryption is indeed fine when all the voters are honest, and really cast their intended votes following the official procedure. But voters might deviate from the honest behavior, for multiple reasons. Some might even be ready to change their votes for money or to break privacy. Then, more advanced protections have to be considered, to really achieve a high privacy level.

For example, as explained by Cortier and Smyth [CS11], if a coalition of voters really wants to know Alice's vote, they can duplicate her encrypted ballot and cast it in their names. Indeed, contrarily to paper-based vote, cloning Alice's vote is usually trivial in electronic voting, as encrypted ballots are stored in the ballot-box, and should be public for verifiability. Therefore, if they are enough people, the bias in the final tally will reveal Alice's vote. The impact of such an attack has been analyzed in [MMR22], and it can be devastating. To avoid such an attack (called the CS-attack), the ballot-box can simply exclude multiple identical votes, but this is under the assumption that one cannot modify Alice's encrypted ballot without altering the content. This requires some non-malleability.

Vote-selling is another major threat, when some voters may have strong incentive (with a reward or external pressures) to vote for someone when being able to prove it later on. Actually, any technique that encrypts the ballot on the client-side, without any modification before storing it in the public ballot-box, is subject to vote-selling attacks, and even at high scale: the client-side code can always be patched in order to reveal the randomness used during encryption, as a receipt to convince anybody of the content of the vote. Similarly to the Benaloh challenge, where the randomness used during encryption is given to the voter to let him verify the *cast-as-intended*, this randomness is indeed enough to prove the content of the ballot sent and stored in the ballot-box to any other party. To avoid vote-selling (called VS-attack), encrypted ballots must be randomized before storage, which needs them to be malleable. This shows how electronic voting makes *a priori* contradictory requirements. In the literature, receipt-freeness prevents VS-attacks, while strong receipt-freeness prevents both VS-attacks and CS-attacks [CCFG16,CFL19].

Randomizable Encryption. The usual approach to get receipt-freeness, without additional interactions, is thus to let the voter encrypt his choice with a randomizable encryption scheme, so that the ballot-box can randomize it before storage/publication. This is possible with the ElGamal encryption scheme. But the voter needs the guarantee that the ballot-box cannot alter his vote.

Signatures on randomizable ciphertexts [BFPV11] was the first attempt to provide non-interactive receipt-freeness: the voter signs (using the Waters signature [Wat05]) his encrypted vote (under ElGamal encryption), and just sends it; then the ballot-box can randomize the ciphertext and adapt the signature, without being able to alter the plaintext. But because of the randomizability of the ballot, it does not preclude CS-attacks. This approach has been improved in [CCFG16,CFL19], with an RCCA-secure encryption scheme, providing the strong receipt-freeness: the voter does not have any receipt to sell his vote and the ballot cannot be replayed without being detectable, with an additional proof of knowledge generated by the voter.

Instead of combining ElGamal encryption with Waters signature, [DPP22] uses the one-time Linearly-Homomorphic Signature ($\mathsf{OT\text{-}LH\text{-}Sign}$) from [LPJY13] to get a more efficient instanciation, with Traceable-CCA security. This notion of signature has also been analyzed in [BF20]. We will use a more compact $\mathsf{OT\text{-}LH\text{-}Sign}$, consisting of a unique group element [HPP20], in the slightly stronger generic bilinear group model (GGM) and algebraic group model (AGM).

Non-Interactive Zero-Knowledge Proofs. But this approach, with randomization of the ballots, is more complex in the homomorphic case, where only the aggregated tally is decrypted. In such a situation, ballots are never decrypted and cannot be individually checked in the clear, whereas a unique fake ballot can make the entire election meaningless. To make them verifiable, in their encrypted form, non-interactive zero-knowledge proofs (NIZKs) are required, to prove some specific properties are satisfied by the text in the clear, without revealing any additional information.

Let us first consider a classical setting: a 1-out-of-N choice (either the choice of one candidate, or one list, among N possibilities). To allow homomorphic tally, one usually encodes the vote into N concatenated bits b_i: $\vec{M} = (b_i)_i \in \mathbb{Z}_p^N$. The vote $\vec{M}$ is then encrypted in a component-wise manner, and one proves $b_i \in \{0, 1\}$, for $i = 1, \ldots, N$, and $\sum b_i = 1$. Such a validity proof is linear in N, for both the size and the generation time. An alternative proof can be $\vec{M} \in \mathcal{S} = \{\vec{V}_1, \ldots \vec{V}_N\} \subseteq \mathbb{Z}_p^N$, where $\vec{V}_i$ is the vote for candidate i ($b_i = 1$, and all other values are 0, to allow homomorphic encrypted tally): this is a unique proof of N-subset membership. More proofs of subset membership, on $\mathbf{A}_k \cdot \vec{M} \in \mathcal{S}_k$ for multiple matrices $\mathbf{A}_k$, can be combined to encompass more complex elections, thanks to the linear property of the encryption scheme. NIZKs of subset membership are the core of homomorphic voting systems, and the main goal of this work.

Such NIZKs exist on ElGamal ciphertexts in groups $\mathbb{G}$, where the Diffie-Hellman assumption holds, or $\mathbb{Z}_p$, when working in the exponents. The most famous use the Fiat-Shamir paradigm on Schnorr-like proofs, to prove disjunctions (OR-proofs). It makes the ballot non-malleable, which helps to exclude CS-attacks but, on the other hand, this is in favor of VS-attacks. The Groth-Sahai methodology [GS08] provides another approach for non-interactive zero-knowledge proofs, that are randomizable, but this is at a very high computational cost (see [CCFG16] for simple 0-1 proofs), which is not reasonable on low-power devices. Most importantly, both previous techniques lead to proofs that have a size and a generation-time linear in the size of the subset $\mathcal{S}$. On the other hand, recent SNARGs (or SNARKs), for *Succinct Non-Interactive Arguments (of Knowledge)* [PHGR13,GGPR13,BCC+16,Gro16], and their zero-knowledge variants, lead to quite efficient proofs for algebraic circuits. However, they suffer two main drawbacks in the setting of electronic voting: (i) while the resulting proofs are quite short and fast to verify, their generation by the prover depends on the complexity of the algebraic circuit. And statements to be proven (as listed above, and in particular disjunctions such as range proofs or proofs of subset membership) lead to quite large algebraic circuits: proof generation-time might get quite high (at least linear in the size of the set $\mathcal{S}$); (ii) the SNARGs methodology requires a complex Common Reference String (CRS). This is a structured tuple that must be generated in a trusted way. Because of the complex structure, it is quite hard to distribute the generation among the electoral board members, and it is not verifiable. Such an approach has been proposed in [LCKO19], using the Groth16 proofs [Gro16], which require a CRS with successive powers and inverses (that are inefficient to distribute) applied on group elements. Without distributed generation, this excludes the fundamental assumption in electronic voting: one only trusts the electoral board as a whole, not any individual party.

Contributions. We exploit linearly-homomorphic signatures [LPJY13,HPP20,HP22,DPP22] to design short and efficient homomorphic quasi-adaptive NIZKs of subset membership on ElGamal ciphertexts, where quasi-adaptive means that the CRS (or the setup) depends on the subset [JR13,KW15]. But contrarily to all the previous methods presented above, that were using signatures on randomizable ciphertexts just for allowing randomization on the server-side, and thus for achieving (strong) receipt-freeness for a mixnet-based system, we additionally use them on the client-side, from multiple signatures generated at the setup phase, by the electoral board, to derive efficient proofs of valid ballots. This can be seen in the same vein as [CFL19], with initial signatures on randomized ballots, but that paper only focused on the cast-as-intended property, and not the validity proof of the ballots, with a costly signature on randomizable ciphertexts, that cannot scale for elections with complex ballots. Our NIZK of subset membership (which is also a signature) has a constant size and takes a constant-time for generation on the voter device, whatever the size of the subset. We stress that when the signing keys and the initial signatures can be efficiently generated in a distributed way, this avoids the need of a trusted third party.

In order to get signatures on randomizable ciphertexts with unlinkability, from linearly-homomorphic signatures ($\mathsf{LH\text{-}Sign}$) on ElGamal ciphertexts, we need *full-fledge* $\mathsf{LH\text{-}Sign}$ [LPJY13,KW15] with tags, to have multiple vector sub-spaces, that cannot be combined. We also need randomizable tags to make signatures unlinkable, whichever is the vector sub-space: Hence, we use Linearly-Homomorphic Signatures with Randomizable Tags ($\mathsf{LH\text{-}Sign\text{-}RTag}$), where the tags target the vector sub-spaces, but still being randomizable and unlinkable, as in [FHS19,HPP20,HP22], in order to keep privacy of the ciphertexts. Signatures for proofs can only be generated for valid ballots, without revealing any information about the votes.

Our contribution is thus the use of $\mathsf{LH\text{-}Sign\text{-}RTag}$ [FHS19,HPP20,HP22] in order to build efficient quasi-adaptive NIZKs of subset membership that are randomizable: they have constant generation-time and constant proof-size. This approach excludes VS-attacks and is compatible with the other improvements [CCFG16,CFL19,DPP22]. In particular, the voter can use a one-time linearly-homomorphic signature ($\mathsf{OT\text{-}LH\text{-}Sign}$) to trace and check his vote after randomization, and append a single Groth-Sahai Diffie-Hellman proof to avoid CS-attacks. This provides strong receipt-freeness. We furthermore finely-tune the SDH-based $\mathsf{LH\text{-}Sign\text{-}RTag}$ [HPP20,HP22] to avoid any unique trusted party, which is hard to obtain with any other similar approach.

Technical Overview. Let us first briefly explain the global idea for proving an ElGamal ciphertext $(C_0 = r \cdot P, \vec{C} = \vec{M} + r \cdot \vec{Z}) \in \mathbb{G}^{n+1}$ actually encrypts a plaintext $\vec{M} \in \mathcal{S} \subset \mathbb{G}^n$, under the encryption key $\vec{Z} \in \mathbb{G}^n$, where $\mathbb{G} = \langle P \rangle$ is a group spanned by a generator P, where the Diffie-Hellman assumption holds.

Thereafter, by applying a matrix $\mathbf{A}_k$ on the second part of the ciphertext $(C_0, \vec{C})$: $\vec{C}' = \mathbf{A}_k \cdot \vec{C} = \mathbf{A}_k \cdot \vec{M} + r \cdot \mathbf{A}_k \cdot \vec{Z}$, one can see $(C_0, \vec{C}')$ as a ciphertext of $\vec{M}' = \mathbf{A}_k \cdot \vec{M}$ under the encryption key $\vec{Z}' = \mathbf{A}_k \cdot \vec{Z}$. The matrix $\mathbf{A}_k$ can be a projection (to focus on some part of the vector) or an aggregation (to sum some components of the vector). Hence, the subset $\mathcal{S}$ can be $\{0 \cdot P, 1 \cdot P\} \subset \mathbb{G}^1$, a larger range $\{0 \cdot P, \ldots, k \cdot P\} \subset \mathbb{G}^1$, or any list of vectors $\{\vec{M}_1, \ldots, \vec{M}_N\} \subset \mathbb{G}^n$ that specifies the admissible aggregations/sums. Multiple proofs can be combined, thanks to their high efficiency.

Let us target on one proof of subset membership, and thus on the set $\mathcal{S} = \{\vec{M}_1, \ldots, \vec{M}_N\}$. The authority first generates $\mathsf{LH\text{-}Sign\text{-}RTag}$ signatures under a verification key VK: $\Sigma_{i,0}$ on $(P_\mathcal{S}, 0, \vec{M}_i)$ and $\Sigma_{i,1}$ on $(0, P, \vec{Z})$, under the common tag τ_i, for $i = 1, \ldots, N$. The first component $P_\mathcal{S}$ is a fixed group element that depends on the set $\mathcal{S}$. It can be set as $P_\mathcal{S} = \mathcal{H}(\mathcal{S})$ where the function $\mathcal{H}$ is assumed to be a full-domain hash function that outputs independent group elements for any new query (modelled as a random oracle onto $\mathbb{G}$). For a choice j, a random combination of the two signatures leads to a valid signature under VK on a ciphertext of $\vec{M} = \vec{M}_j$: $\Sigma_0 = \Sigma_{j,0} + r \cdot \Sigma_{j,1}$ is a valid signature on $(P_\mathcal{S}, C_0 = r{\cdot}P, \vec{C} = r{\cdot}\vec{Z}+\vec{M}_j) = (P_\mathcal{S}, 0, \vec{M}_j)+r{\cdot}(0, P, \vec{Z})$ under $\tau = \tau_j$. Note that the first components $P_\mathcal{S}$ and 0, later checked with respect to $P_\mathcal{S}$ for Σ_0, imply the coefficient 1 on $(P_\mathcal{S}, 0, \vec{M}_j)$ in the combination. We stress that without tags, one could not prevent someone to encrypt $(K+1) \cdot \vec{M}_1 - K \cdot \vec{M}_2$, for any K of its choice, which is not considered a legitimate ciphertext. One also keeps $\Sigma_1 = \Sigma_{j,1}$ as a valid signature on the *randomizer* $(0, D_0 = P, \vec{D} = \vec{Z})$ under the tag $\tau = \tau_j$, for further randomization.

We have dropped the index j in the signatures (Σ_0, Σ_1) and the tag τ, as there is a unique pair of ciphertexts that remains: $\mathcal{C} = (C_0, \vec{C})$ of $\vec{M}_j$ and $\mathcal{D} = (D_0, \vec{D})$ of $\vec{0}$. They do not reveal information about j, under the DDH assumption (ElGamal encryption), but the signatures (Σ_0, Σ_1) are valid under τ_j only, which reveals j. Hence the need of randomizable tags to make the final tag τ and the signatures (Σ_0, Σ_1) unlinkable to τ_j, and thus independent from the choice j.

Linearly-Homomorphic Signature with Randomizable Tags. The rest of the paper will explain how to modify the above $(\mathcal{C}, \mathcal{D})$ and $(\Sigma_0, \Sigma_1, \tau)$ before sending them in order to keep vote-privacy, using the randomizable tags. There are two $\mathsf{LH\text{-}Sign\text{-}RTag}$ candidates in the literature.

First, the FHS signature [FHS19], in a type III pairing-friendly setting $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, p, P, \hat{P}, e)$, also presented as a signature on randomizable ciphertexts [BF20]: a tag is $\tau = (\tau_1 = 1/t \cdot P, \tau_2 = 1/t \cdot \hat{P})$, for a scalar $t \xleftarrow{\$} \mathbb{Z}_p$, the signature of $\vec{M} = (M_k) \in \mathbb{G}^n$ is $\Sigma = t \cdot (\sum s_k \cdot M_k) \in \mathbb{G}$, under $\mathsf{VK} = (\hat{P}_k = s_k \cdot \hat{P})_k$, that can both be verified by $e(P, \tau_2) = e(\tau_1, \hat{P})$ and $e(\Sigma, \tau_2) = \prod e(M_k, \hat{P}_k)$. One can easily randomize τ and adapt the signature Σ, in a perfectly unlinkable way. Hence, the privacy relies on the ElGamal encryption scheme only. From a CRS of linear length in the size of the subset $\mathcal{S}$ (with the $(\Sigma_{i,0}, \Sigma_{i,1}, \tau_i)_i$ for all the $\vec{M}_i \in \mathcal{S}$), proofs of subset membership are thereafter both size and time efficient (independent of the size of $\mathcal{S}$) as the voter only has to generate $\mathcal{C} = (C_0, \vec{C})$ for appropriate choice $\vec{M}_j$, and to randomize $(\Sigma_0, \Sigma_1, \tau) \in \mathbb{G}^3 \times \hat{\mathbb{G}}$, keeping $\mathcal{D} = (P, \vec{Z})$ unchanged. As the tags are self-verifiable (without any additional proof) signatures are very compact.

While the CRS has a similar size to the one for SNARGs (linear in the size of the algebraic circuit), SNARGs also have a linear generation-time of the proof, whereas ours is constant-time. Unfortunately, the setup that generates the CRS suffers the same drawback of being hard to distribute, because of the modular inverses to be computed.

Hence, in the following, we consider the second $\mathsf{LH\text{-}Sign\text{-}RTag}$ candidate, with Square Diffie-Hellman tags $\tau = (P, t \cdot P, t^2 \cdot P)$, for $t \xleftarrow{\$} \mathbb{Z}_p$ [HPP20,HP22]. The tags will need additional proofs to be verifiable, but the CRS can be efficiently generated in a distributed way.

2 Preliminaries

Computational Assumptions. Our security analysis will be performed in the Generic Group Model (GGM) and the Algebraic Group Model (AGM) [FKL18], where the adversary can only make generic operations on the group (and pairing evaluations). Hence, any new generated group element comes as a linear combination of the input group elements. The former GGM is a slightly stronger model than the latter AGM, as encodings of group elements can even be chosen at random. But in both cases, any group element provided by the adversary comes with the explicit coefficients of the linear combination of the input elements. This will provide the simulation extractability of our proofs. We will use the following classical assumptions in a group $\mathbb{G}$ of prime order p, for any $P \in \mathbb{G}$:

Discrete Logarithm (DL) Assumption. Given $(P, U = x \cdot P)$, for $x \xleftarrow{\$} \mathbb{Z}_p$, it is computationally hard to recover x;

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

Square Discrete Logarithm (SDL) Assumption. Given $(P, U = x \cdot P, V = x^2 \cdot P)$, for $x \xleftarrow{\$} \mathbb{Z}_p$, it is computationally hard to recover x;

Decisional Square Diffie-Hellman (DSDH) Assumption. For $x, y \xleftarrow{\$} \mathbb{Z}_p$, distributions $\mathcal{D}_{\mathsf{sdh}} = \{(P, x \cdot P, x^2 \cdot P)\}$ and $\mathbb{G}^3_\$ = \{(P, x \cdot P, y \cdot P)\}$ are computationally hard to distinguish.

The SXDH assumption claims that the DDH assumption holds in both groups $\mathbb{G}$ and $\hat{\mathbb{G}}$, when we are in a type III pairing-friendly setting $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, p, P, \hat{P}, e)$, with e a bilinear map from $\mathbb{G} \times \hat{\mathbb{G}}$ into the target group $\mathbb{G}_T$.

Groth-Sahai Proofs. The Groth-Sahai methodology [GS08] is well-known for zero-knowledge and randomizable proofs of pairing-product relations. It is not appropriate for complex relations, but efficient enough for short statements, such as Diffie-Hellman tuples $(H, U = t \cdot H, R, V = t \cdot R) \in \mathbb{G}^4$, or SDH tuples: one defines the CRS as a random tuple $(\hat{V}_{1,1}, \hat{V}_{1,2}, \hat{V}_{2,1}, \hat{V}_{2,2}) \in \hat{\mathbb{G}}^4$, likely a non-Diffie-Hellman tuple. From the witness $t \in \mathbb{Z}_p$, one commits it, for $\mu \xleftarrow{\$} \mathbb{Z}_p$:

$$\mathsf{com} = (\hat{C} = t \cdot \hat{V}_{2,1} + \mu \cdot \hat{V}_{1,1}, \hat{D} = t \cdot \hat{V}_{2,2} + \mu \cdot \hat{V}_{1,2}),$$

and sets $\Theta = \mu \cdot H$ and $\Psi = \mu \cdot R$, which satisfy

$$\begin{aligned} e(H, \hat{C}) &= e(U, \hat{V}_{2,1}) \cdot e(\Theta, \hat{V}_{1,1}), & e(H, \hat{D}) &= e(U, \hat{V}_{2,2}) \cdot e(\Theta, \hat{V}_{1,2}), \\ e(R, \hat{C}) &= e(V, \hat{V}_{2,1}) \cdot e(\Psi, \hat{V}_{1,1}), & e(R, \hat{D}) &= e(V, \hat{V}_{2,2}) \cdot e(\Psi, \hat{V}_{1,2}). \end{aligned}$$

The proof is thus $\mathsf{proof} = (\Theta, \Psi) \in \mathbb{G}^2$, on the tuple (H, U, R, V) and the commitment $\mathsf{com} = (\hat{C}, \hat{D})$. To verify it, instead of checking the four equations independently, which require 12 pairing evaluations, one can apply a batch verification [BFI+10] which just consists of 3 pairing evaluations. The soundness is perfect, while the zero-knowledge property relies on the SXDH assumption.

SDH Tags and Properties. We first recall some properties for the SDH tuples: the unlinkability and the non-miscibility studied in [HPP20, Full version]. As shown in [HPP20,HP22], the DSDH and the DDH assumptions imply the unlinkability of two SDH tuples. Furthermore, under the SDL assumption, SDH tuples with different scalars cannot be mixed under known linear combinations:

Proposition 1 (Unlinkability). *For any $P \in \mathbb{G}$, the* DDH *and* DSDH *assumptions imply the indistinguishability of $\mathcal{D}_0$ and $\mathcal{D}_1$, whith $U \xleftarrow{\$} \mathbb{G}, x, y \xleftarrow{\$} \mathbb{Z}_p$:*

$$\mathcal{D}_0 = \{(P, x \cdot P, x^2 \cdot P, U, x \cdot U, x^2 \cdot U)\} \qquad \mathcal{D}_1 = \{(P, x \cdot P, x^2 \cdot P, U, y \cdot U, y^2 \cdot U)\}$$

Proposition 2 (Non-Miscibility). *Given n SDH tuples $(P, U_i = x_i \cdot P, V_i = x_i \cdot U_i)$, for any generator P, but random $x_i \xleftarrow{\$} \mathbb{Z}_p$, outputting $(\alpha_i)_{i=1,\ldots,n}$ such that $(H = \sum \alpha_i \cdot P, U = \sum \alpha_i \cdot U_i, V = \sum \alpha_i \cdot V_i)$ is an SDH tuple, with at least two non-zero coefficients α_i, is computationally hard under the* SDL *assumption.*

However, verifying an SDH tuple requires an additional proof, which can be done with the above Groth-Sahai methodology, with the CRS $(\hat{V}_{1,1}, \hat{V}_{1,2}, \hat{V}_{2,1}, \hat{V}_{2,2})$. Given an SDH tuple $(H, U = x \cdot H, V = x \cdot U)$ in $\mathbb{G}$, knowing the witness $x \in \mathbb{Z}_p$, one first commits it, for a random $\nu \xleftarrow{\$} \mathbb{Z}_p$:

$$\mathsf{com} = (\hat{C} = x \cdot \hat{V}_{2,1} + \nu \cdot \hat{V}_{1,1}, \hat{D} = x \cdot \hat{V}_{2,2} + \nu \cdot \hat{V}_{1,2}),$$

and one sets $\mathsf{proof} = (\Theta = \nu \cdot H, \Psi = \nu \cdot U)$, which satisfies the four above equalities, where $R = U$. This proven tuple $(\vec{\tau} = (H, U, V), \mathsf{com} = (\hat{C}, \hat{D}), \mathsf{proof} = (\Theta, \Psi))$, which consists of 5 elements in $\mathbb{G}$ and 2 elements in $\hat{\mathbb{G}}$, is randomizable: one can publicly update H in both $\vec{\tau}$ and proof (with a chosen multiplicative factor), and ν in both com and proof (without knowing it, but just the additional value ν'), making the new proven tuple unlinkable to the initial one, from the unlinkablity of the tuples and the zero-knowledge property of the proof.

But for correct combinations, we need a stricter notion of equivalent (valid) tags.

Definition 3 (Equivalent SDH Tags). *An SDH tag will be a tuple $\mathsf{Tag} = (\vec{\tau}, \mathsf{proof}, \mathsf{com})$, with valid proof, and two tags will be said equivalent if they not only are for the same scalar x, but also for the same commitment* com.

Proposition 4 (Linearity). *Given n equivalent SDH tags, with $\vec{\tau}_i = (H_i, U_i = x \cdot H_i, V_i = x^2 \cdot H_i)$, for the same $x \in \mathbb{Z}_p$, their proofs $\mathsf{proof}_i = (\Theta_i = \nu \cdot H_i, \Psi_i = \nu \cdot U_i)$, for the same ν, and thus the common commitment $\mathsf{com} = (\hat{C} = x \cdot \hat{V}_{2,1} + \nu \cdot \hat{V}_{1,1}, \hat{D} = x \cdot \hat{V}_{2,2} + \nu \cdot \hat{V}_{1,2})$, for any linear combination $\vec{\tau} = \sum \alpha_i \cdot \vec{\tau}_i$, $\mathsf{proof} = (\Theta = \sum \alpha_i \cdot \Theta_i, \Psi = \sum \alpha_i \cdot \Psi_i)$ is a valid proof for com.*

Actually, $\vec{\tau} = (H, U, V)$ with $H = \sum \alpha_i \cdot H_i$, hence the validity of proof. The proof and the commitment can thereafter be randomized, with a new $\nu \overset{\$}{\leftarrow} \mathbb{Z}_p$.

3 Linearly-Homomorphic Signatures

Linearly-homomorphic signatures ($\mathsf{LH\text{-}Sign}$) were introduced in [BFKW09], to sign vector sub-spaces. They allow to combine any signatures on vectors, so that one can derive the signature of a linear combination of the already signed vectors, but nothing else. Then, Libert *et al.* [LPJY13] proposed a linearly-homomorphic signature scheme, that is furthermore structure-preserving. More recently, Hébant *et al.* [HPP20] adapted their scheme for a simpler One-Time Linearly-Homomorphic signature ($\mathsf{OT\text{-}LH\text{-}Sign}$), proven in the GGM, together with the family of Square Diffie-Hellman tags, to provide anonymity properties [HP22]. It can be seen as a Linearly-Homomorphic signature with Randomizable Tags ($\mathsf{LH\text{-}Sign\text{-}RTag}$), where multiple sub-spaces can be signed independently (from now, referred as the SDH signature). Alternatively, the above Fuchsbauer *et al.* scheme [FHS19,BF20] can also be used (later referred as the FHS signature). The latter provides better efficiency and compactness, but the former allows efficient distributed generation of the CRS, which is more important for our voting application.

3.1 One-Time Linearly-Homomorphic Signature ($\mathsf{OT\text{-}LH\text{-}Sign}$)

We first recall the simplified $\mathsf{OT\text{-}LH\text{-}Sign}$, derived from Libert *et al.* [LPJY13], proven secure in the GGM [FHS19,HPP20,BF20], with messages in $\mathcal{M} = \mathbb{G}^n$:

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

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

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

$\mathsf{DerivSign}(\mathsf{vk}, (w_i, \vec{M}_i, \sigma_i)_{i=1}^{\ell})$**:** Given a public key vk and ℓ tuples of weights $w_i \in \mathbb{Z}_p$ and signed messages $\vec{M}_i$ in σ_i, it outputs the signature $\sigma = \sum_{i=1}^{\ell} w_i \cdot \sigma_i$ of the vector $\vec{M} = \sum_{i=1}^{\ell} w_i \cdot \vec{M}_i$;

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

The correctness can be easily checked, thanks to the bilinearity of pairing e. Unforgeability holds in the GGM [FHS19,HPP20]: one can only derive signatures on messages in the span of the already signed messages. The GGM provides the simulator with the coefficients of the linear combination, while the programmable encodings allow to answer signing queries, in the chosen-message scenario. To allow multiple subspaces, one uses tags. Combinations should then only be possible between messages signed under the same tag.

3.2 SDH-Based Linearly-Homomorphic Signature

Thanks to the non-miscibility of the SDH tags (see Proposition 2), one can transform any One-Time Linearly-Homomorphic signature ($\mathsf{OT\text{-}LH\text{-}Sign}$) into a (full-fledge) Linearly-Homomorphic Signature. The unlinkability of the SDH tags (see Proposition 1) make them randomizable tags ($\mathsf{LH\text{-}Sign\text{-}RTag}$). But let us directly describe our signature scheme with messages in $\mathcal{M} = \mathbb{G}^n$:

$\mathsf{Setup}(1^\kappa)$**:** Given κ, it outputs param, that contains a pairing-friendly setting $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, p, P, \hat{P}, e)$ and a random tuple $(\hat{V}_{1,1}, \hat{V}_{1,2}, \hat{V}_{2,1}, \hat{V}_{2,2}) \overset{\$}{\leftarrow} \hat{\mathbb{G}}^4$;

$\mathsf{Keygen}(\mathsf{param}, n)$**:** Given param and an integer n, it generates $\mathsf{sk} = \vec{s} \overset{\$}{\leftarrow} \mathbb{Z}_p^{n+3}$, sets $\mathsf{vk} = \vec{s} \cdot \hat{P} = (\hat{P}_i = s_i \cdot \hat{P})_{i=1}^{n+3} \in \hat{\mathbb{G}}^{n+3}$, and outputs the key pair $(\mathsf{sk}, \mathsf{vk})$;

$\mathsf{NewTag}(\mathsf{param})$**:** Generates a verifiable tag $\mathsf{Tag} = (\vec{\tau} = (P, t \cdot P, t^2 \cdot P), \mathsf{proof}, \mathsf{com})$, for random scalars $t, \nu \overset{\$}{\leftarrow} \mathbb{Z}_p$, used in proof and com;

$\mathsf{Sign}(\mathsf{sk}, \mathsf{Tag}, \vec{M})$**:** Given a signing key sk, a verifiable tag $\mathsf{Tag} = (\vec{\tau}, \mathsf{proof}, \mathsf{com})$, and a vector-message $\vec{M} = (M_i)_i \in \mathbb{G}^n$, it outputs the signature $\sigma = \langle \vec{s}, \vec{M} \| \vec{\tau} \rangle \in \mathbb{G}$, where the full message is $\vec{M} \| \vec{\tau} = (M_1, \ldots, M_n, \tau_1, \tau_2, \tau_3) \in \mathbb{G}^{n+3}$;

$\mathsf{DerivSign}(\mathsf{vk}, (w_i, \mathsf{Tag}_i, \vec{M}_i, \sigma_i)_{i=1}^{\ell})$**:** Given a public key vk and ℓ tuples of weights $w_i \in \mathbb{Z}_p$ and signed messages $\vec{M}_i$ in σ_i, under equivalent tags Tag_i, it outputs the signature $\sigma = \sum w_i \cdot \sigma_i$, on the vector $\vec{M} = \sum_{i=1}^{\ell} w_i \cdot \vec{M}_i$, valid under the equivalent tag Tag' with $\vec{\tau}' = \sum w_i \cdot \vec{\tau}_i$, and adapted proof proof', but the same commitment com;

$\mathsf{Verif}(\mathsf{vk}, \mathsf{Tag}, \vec{M}, \sigma)$**:** Given a verification key vk, a verifiable tag Tag, a vector-message $\vec{M}$ and a signature σ, it outputs 1 if $e(\sigma, \hat{P}) = \prod_{i=1}^{n} e(M_i, \hat{P}_i) \times \prod_{i=1}^{3} e(\tau_i, \hat{P}_{n+i})$ and the tag Tag is valid, and 0 otherwise.

3.3 Distributed Generation

The main advantage of our proofs using SDH signatures, compared to the FHS signatures or SNARGs, is the possible distributed setup, key generation, tag generation, and signatures, among multiple users $(\mathcal{U}_k)_k$:

$\mathsf{Setup}(1^\kappa, k)$**:** They all agree on the bilinear setting $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, p, P, \hat{P}, e)$. User $\mathcal{U}_k$ chooses and sends random points $\hat{V}_{1,1,k}, \hat{V}_{1,2,k}, \hat{V}_{2,1,k}, \hat{V}_{2,2,k} \overset{\$}{\leftarrow} \hat{\mathbb{G}}$. This leads to the global verification points $\hat{V}_{1,1} = \sum_k \hat{V}_{1,1,k}$, $\hat{V}_{1,2} = \sum_k \hat{V}_{1,2,k}$, $\hat{V}_{2,1} = \sum_k \hat{V}_{2,1,k}$, $\hat{V}_{2,2} = \sum_k \hat{V}_{2,2,k}$, to complete param.

$\mathsf{Keygen}(\mathsf{param}, n, k)$**:** Given the public parameters param, $\mathcal{U}_k$ randomly chooses $s_{i,k} \overset{\$}{\leftarrow} \mathbb{Z}_p$, for $i = 1, \ldots, n+3$. $\mathcal{U}_k$ computes and sends $\mathsf{vk}_k = (\hat{P}_{i,k} = s_{i,k} \cdot \hat{P})_{i=1}^{n+3}$. This leads to the global verification key $\mathsf{vk} = \sum_k \mathsf{vk}_k$, while $\mathcal{U}_k$ keeps its signing key share $\mathsf{sk}_k = (\mathsf{sk}_{i,k} = s_{i,k})_{i=1}^{n+3}$.

$\mathsf{NewTag}(\mathsf{param})$**:** each user $\mathcal{U}_k$ chooses random $t_k, \nu_k \overset{\$}{\leftarrow} \mathbb{Z}_p$:

1. $\mathcal{U}_k$ computes and sends $\mathsf{com}_k = (\hat{C}_k = t_k \cdot \hat{V}_{2,1} + \nu_k \cdot \hat{V}_{1,1}, \hat{D}_k = t_k \cdot \hat{V}_{2,2} + \nu_k \cdot \hat{V}_{1,2})$, and $U_k = t_k \cdot P$, $\Theta_k = \nu_k \cdot P$;
2. $\mathcal{U}_k$ computes $U = \sum_k U_k$, $\Theta = \sum_k \Theta_k$, and sends $V_k = t_k \cdot U$, $\Psi_k = \nu_k \cdot U$.

This allows to compute $V = \sum_k V_k$, $\Psi = \sum_k \Psi_k$, and $\hat{C} = \sum_k \hat{C}_k$, $\hat{D} = \sum_k \hat{D}_k$. This leads to $\mathsf{Tag} = (\vec{\tau} = (P, U, V), \mathsf{proof} = (\Theta, \Psi), \mathsf{com} = (\hat{C}, \hat{D}))$, on $t = \sum t_k$ and $\nu = \sum \nu_k$.

$\mathsf{Sign}(\mathsf{sk}_k, \mathsf{Tag}, \vec{M} = (M_i)_i)$**:** Given a signing key share $\mathsf{sk}_k = (s_{i,k})_i$, a tag $\mathsf{Tag} = (\vec{\tau}, \mathsf{proof}, \mathsf{com})$ and a vector-message $\vec{M} = (M_i)_i \in \mathbb{G}^n$, one outputs the signature share $\sigma_k = \sum_{i=1}^{n} s_{i,k} \cdot M_i + \sum_{i=1}^{3} s_{n+i,k} \cdot \tau_i \in \mathbb{G}$. From those shares, one can compute $\sigma = \sum_k \sigma_k$.

The correctness can easily be verified, and parallelizing some steps, to generate the global parameters param, several tags, and signatures on pre-determined messages, a three-round protocol is enough, with public communications, in the honest-but-curious setting. No additional proofs are required in the malicious setting, as each step is already verifiable: (Θ_k, Ψ_k) is a DH proof on (P, U_k, U, V_k) for the commitment com_k, and σ_k is a valid signature of $(\mathsf{Tag}, \vec{M})$ under vk_k. Additional extractable commitments on every sent value allows perfect simulation even against adaptive adversaries: from the committed values, the simulator can generate all the contributions of the honest players so that the final outcome corresponds to any pre-defined values. Such an extractable commitment on a value x can be done with $H(x, r)$, for a large enough random r, in the random oracle model. Because of the linearity of the operations, they essentially all run a full setup on their own. This is a quite simple interactive process.

4 New Efficient NIZK of Subset Membership

We now detail how one can generate efficient randomizable NIZK of subset membership on ElGamal ciphertexts, using SDH signatures, following the general approach presented in the technical overview, with some optimizations due to the SDH tags. We thereafter detail how to adapt [CCFG16,DPP22] to get strong receipt-freeness.

4.1 Proof of Subset Membership for ElGamal Ciphertexts

We first focus on the setup and then on the ciphertext generation with the proof that the plaintext $\vec{M}$ is in the subset $\mathcal{S} = \{\vec{M}_1, \ldots, \vec{M}_N\} \subset \mathbb{G}^n$: we use a type III pairing-friendly setting $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, p, P, \hat{P}, e)$, with an ElGamal encryption key $\vec{Z} = \vec{z} \cdot P$ (note that $\vec{z} \stackrel{\$}{\leftarrow} \mathbb{Z}_p^n$ can also be generated in a distributed way).

Setup: CRS Generation. Using the above distributed algorithms, one generates signature keys, $\mathsf{SK} = \vec{s} \stackrel{\$}{\leftarrow} \mathbb{Z}_p^{n+4}$ and $\mathsf{VK} = \vec{\hat{S}} \leftarrow \vec{s} \cdot \hat{P} \in \hat{\mathbb{G}}^{n+4}$, as well as N tags $\mathsf{Tag}_i = (\vec{\tau}_i, \mathsf{proof}_i, \mathsf{com}_i)$, for $\vec{\tau}_i = (\tau_{i,1} = P, \tau_{i,2} = t_i \cdot P, \tau_{i,3} = t_i^2 \cdot P)$, with $t_i \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$, for $i = 1, \ldots, N$, with the validity proofs $\mathsf{proof}_i \in \mathbb{G}^2$ and commitments $\mathsf{com}_i \in \hat{\mathbb{G}}^2$ (on t_i, for random ν_i), and N pairs of signatures, for $\vec{M}_i \in \mathcal{S}$:

$$\Sigma_{i,0} = \mathsf{Sign}(\mathsf{SK}, \mathsf{Tag}_i, (P_{\mathcal{S}}, 0, \vec{M}_i)) = \langle (P_{\mathcal{S}}, 0, \vec{M}_i, \tau_{i,2}, \tau_{i,3}), \mathsf{SK} \rangle \in \mathbb{G}$$
$$\Sigma_{i,1} = \mathsf{Sign}(\mathsf{SK}, \mathsf{Tag}_i, (0, P, \vec{Z})) = \langle (0, P, \vec{Z}, \tau_{i,2}, \tau_{i,3}), \mathsf{SK} \rangle \in \mathbb{G}$$

We note the optimization, where $\tau_{i,1} = P$ is a known constant, so there is no need to provide it: the length of the vectors is $n+4$ only. The verification key VK, the tags $(\mathsf{Tag}_i)_i$ and the pairs of signatures $(\Sigma_{i,0}, \Sigma_{i,1})_i$ constitute the CRS.

Proven Ciphertext Generation. The encryptor can generate a random ciphertext of $\vec{M}_j$ as $\mathcal{C} = (C_0, \vec{C}) = (r \cdot P, \vec{M}_j + r \cdot \vec{Z}) = (0, \vec{M}_j) + r \cdot (P, \vec{Z})$, and the random randomizer $\mathcal{D} = (D_0, \vec{D}) = (s \cdot P, s \cdot \vec{Z}) = s \cdot (P, \vec{Z})$, for random $r, s \stackrel{\$}{\leftarrow} \mathbb{Z}_p$, together with the associated signatures $\Sigma_0 = \Sigma_{j,0} + r \times \Sigma_{j,1}$ of $(P_{\mathcal{S}}, C_0, \vec{C})$ and $\Sigma_1 = s \cdot \Sigma_{j,1}$ of $(0, D_0, \vec{D})$ under the equivalent tags $\vec{\tau}'_0 = (r+1) \cdot \vec{\tau}_j$ and $\vec{\tau}'_1 = s \cdot \vec{\tau}_j$ for the key VK. The tags have already been randomized, and one can randomize com' and adapt proof'_0, proof'_1, with an additional random $\nu \stackrel{\$}{\leftarrow} \mathbb{Z}_p$. It then sends the twin-ciphertexts $(\mathcal{C}, \mathcal{D}) \in \mathbb{G}^{2n+2}$, the twin-signatures $(\Sigma_0, \Sigma_1) \in \mathbb{G}^2$ and the twin-tags $(\vec{\tau}'_0, \mathsf{proof}'_0, \vec{\tau}'_1, \mathsf{proof}'_1, \mathsf{com}') \in \mathbb{G}^8 \times \hat{\mathbb{G}}^2$. The first components of the tags are $C_0 + P$ and D_0 respectively, there is no need to duplicate them.

Randomization. The receiver can randomize the ciphertext into $\mathcal{C}' = (C'_0, \vec{C}') = \mathcal{C} + r' \cdot \mathcal{D}$, with $r' \stackrel{\$}{\leftarrow} \mathbb{Z}_p$, adapt the tag $\vec{\tau}' = \vec{\tau}'_0 + r' \cdot \vec{\tau}'_1$, and randomize it with $\nu' \stackrel{\$}{\leftarrow} \mathbb{Z}_p$ in $\mathsf{proof}', \mathsf{com}'$, and adapt the signature $\Sigma' = \Sigma_0 + r' \cdot \Sigma_1$. In the end, one only stores $\mathcal{C}' = (C'_0, \vec{C}') \in \mathbb{G}^{n+1}$ and the proof $(\Sigma', \mathsf{Tag}' = (\vec{\tau}', \mathsf{proof}', \mathsf{com}')) \in \mathbb{G}^5 \times \hat{\mathbb{G}}^2$, which is independent of N.

Verifiable Tally. The valid ballots can be aggregated by summing component-wise all the ciphertexts $\{(C'_0, \vec{C}')\}$ from the public ballot-box into a global ciphertext $\mathcal{T} = (T_0, \vec{T})$, which decrypts to the tally using the decryption key $\vec{z}$, with zero-knowledge proofs of valid decryption (*à la Schnorr* with the Fiat-Shamir paradigm).

More Constraints. Our approach is thus quite efficient for a subset constraint, whatever the size of the subset. And multiple constraints on ciphertexts $(C_0, \mathbf{A}_k \cdot \vec{C})$ under the keys $\vec{Z}_k = \mathbf{A}_k \cdot \vec{Z}$ in sets $\mathcal{S}_k$ can be combined: each one needs twin-tags and twin-signatures, sent by the voter (10 elements of $\mathbb{G}$ and 2 elements of $\hat{\mathbb{G}}$), but only one tag and one signature stored in the ballot-box (5 elements of $\mathbb{G}$ and 2 elements of $\hat{\mathbb{G}}$). More complex formats of the ballots can be modelled as subset constraints. One just has to take care of the first component $P_{\mathcal{S}_k}$ that must be specific to $\mathcal{S}_k$, to avoid combinations between elements in the CRS that are for different proofs. Any successful illegal combination would break the DL assumption in the algebraic group model and the random oracle model

Examples. The 0-1 choices (each i-th box must be a 0-1 choice) is a subset constraint, with $\mathcal{S}_i = \{0 \cdot P, 1 \cdot P\} \subset \mathbb{G}^1$, for each i. This allows to prove the validity of the 0-1 encryption in (C_0, C_i) under the ElGamal key Z_i. However, this requires n subset membership proofs, for sets of size 2. And the less constraints there are, the more interesting is our approach, even if the sets get larger. Hence, one can consider $\mathcal{S} = \{0,1\}^n$, and thus only one subset membership proof, with a set of size 2^n, if n is not too large. But then the setup becomes exponential in n. One can also split n to have k parallel subset membership proofs of length $2^{n/k}$. We however stress that even for any quite large (possibly exponential) CRS, the generation of the proof by the encryptor is constant-time.

For a more complex election, where one can check at most K boxes among N, when all boxes being 0-1, as above, one can consider the additional proof of subset membership with $\mathcal{S} = \{0 \cdot P, \dots, K \cdot P\} \subset \mathbb{G}^1$ for the ciphertext $(C_0, \sum C_i)$ under the key $Z = \sum Z_i$.

4.2 Proofs on Randomizable Ciphertexts

Interestingly, this approach is compatible with [CCFG16,DPP22] to get strong receipt-freeness: an encryptor can restrict transformations on his ElGamal ciphertext thanks to an $\mathsf{OT\text{-}LH\text{-}Sign}$ in $\mathbb{G}^{n+2}$, for a personal verification key vk, by signing both the ciphertext $(P, C_0, \vec{C})$ and the randomizer $(0, D_0, \vec{D})$ in (σ_0, σ_1), without any need of tags, as this is the only vector sub-space with this signing key sk: any new signature σ on some $(P, C'_0, \vec{C}')$, valid under vk, is necessarily a randomization of $(C_0, \vec{C})$. This provides receipt-freeness. We stress that $(\mathsf{sk}, \mathsf{vk})$ is an individual pair of keys that is owned by the signer, contrarily to $(\mathsf{SK}, \mathsf{VK})$ that was generated during the setup for the CRS.

According to [CCFG16], one can add a Groth-Sahai proof of Diffie-Hellman tuple for $(P, T = \mathcal{H}(\mathsf{vk}), C_0 = r \cdot P, W_0 = r \cdot T)$, as suggested in [CCFG16]. But in our case, the RCCA-security requires the AGM to extract r from such a valid Diffie-Hellman tuple with a truly random T. The randomness r then allows the decryption. We also have to do it for $(P, T = \mathcal{H}(\mathsf{vk}), D_0 = s \cdot P, W_1 = s \cdot T)$, for future randomization: for random ν_0, ν_1, one generates $\mathsf{com}_0 = (\hat{C}_0 = r \cdot \hat{V}'_{2,1} + \nu_0 \cdot \hat{V}'_{1,1}, \hat{D}_0 = r \cdot \hat{V}'_{2,2} + \nu_0 \cdot \hat{V}'_{1,2})$, $\mathsf{com}_1 = (\hat{C}_1 = s \cdot \hat{V}'_{2,1} + \nu_1 \cdot \hat{V}'_{1,1}, \hat{D}_1 = s \cdot \hat{V}'_{2,2} + \nu_1 \cdot \hat{V}'_{1,2})$, and the proofs $\mathsf{proof}_0 = (\Theta_0 = \nu_0 \cdot P, \Psi_0 = \nu_0 \cdot T)$, $\mathsf{proof}_1 = (\Theta_1 = \nu_1 \cdot P, \Psi_1 = \nu_1 \cdot T)$, that can later be combined by into $\mathsf{com} = \mathsf{com}_0 + r' \cdot \mathsf{com}_1$ and $\mathsf{proof} = \mathsf{proof}_0 + r' \cdot \mathsf{proof}_1$ by the receiver(s) to verify $W = W_0 + r' \cdot W_1 = r'' \cdot T$, with unknown scalars $r'' = r + r's$ and $\nu'' = \nu_0 + r'\nu_1$, after having also randomized the ciphertext $C'_0 = C_0 + r' \cdot D_0$ and $\vec{C}' = \vec{C} + r' \cdot \vec{D}$. We stress that an independent CRS $(\hat{V}'_{1,1}, \hat{V}'_{1,2}, \hat{V}'_{2,1}, \hat{V}'_{2,2})$ is used here. The same as for the SDH proofs could be used, but the separation will clarify the security analysis.

5 Homomorphic Voting System: Efficiency and Security

We first recap the communications and computations, in the N-subset case, for ballots where n boxes could be checked: $\vec{M} \in \mathcal{S} \subset \mathbb{G}^n$. Thereafter, we explain the security properties. We stress that we target a homomorphic voting system.

Communication. The full public information generated at the setup time, in a possibly distributed way, consists of $\mathsf{VK} \in \hat{\mathbb{G}}^{n+4}$, $(\Sigma_{i,0}, \Sigma_{i,1})_i \in \mathbb{G}^{2N}$, with the tags, but without the constant P, $(\tau_{i,2}, \tau_{i,3})_i \in \mathbb{G}^{2N}$, together with their proofs and commitments in $(\mathbb{G}^2 \times \hat{\mathbb{G}}^2)^N$, and the two Groth-Sahai CRS in $\hat{\mathbb{G}}^4$. The CRS thus consists of $6N$ elements from $\mathbb{G}$ and $n + 2N + 12$ elements from $\hat{\mathbb{G}}$.

From the CRS, the voter sends the twin-ciphertexts $(\mathcal{C}, \mathcal{D}) \in \mathbb{G}^{2n+2}$, the twin-validity-proofs $(\Sigma_0, \Sigma_1, \vec{\tau}'_0, \mathsf{proof}'_0, \vec{\tau}'_1, \mathsf{proof}'_1, \mathsf{com}') \in \mathbb{G}^{10} \times \hat{\mathbb{G}}^2$, the twin-signatures $(\mathsf{vk}, \sigma_0, \sigma_1) \in \hat{\mathbb{G}}^{n+2} \times \mathbb{G}^2$, and the twin-user-proofs $(W_0, W_1, \mathsf{proof}_0, \mathsf{proof}_1, \mathsf{com}_0, \mathsf{com}_1) \in \mathbb{G}^6 \times \hat{\mathbb{G}}^4$, using above notations. We thus globally have $2n + 20$ elements from $\mathbb{G}$ and $n + 8$ elements from $\hat{\mathbb{G}}$ in the twin-ballot.

The ballot-box stores the randomized ciphertext $\mathcal{C}' \in \mathbb{G}^{n+1}$, together with the validity proof $(\Sigma'', \vec{\tau}'', \mathsf{proof}'', \mathsf{com}'') \in \mathbb{G}^5 \times \hat{\mathbb{G}}^2$, as well as the user-signature $(\mathsf{vk}, \sigma) \in \hat{\mathbb{G}}^{n+2} \times \mathbb{G}$, and the user-proof $(W'', \mathsf{proof}, \mathsf{com}) \in \mathbb{G}^3 \times \hat{\mathbb{G}}^2$. We thus globally have $n + 10$ elements from $\mathbb{G}$ and $n + 6$ elements from $\hat{\mathbb{G}}$ for each randomized ballot.

We can use the type III pairing-friendly curve BLS12-381 [BLS03], as in all the zk-SNARKs applications: $\mathbb{G}$ group elements are encoded on 48 bytes, while $\hat{\mathbb{G}}$ group elements are encoded on 96 bytes. For $N = n = 25$ (in the 1-out-of-n case), the public information is a bit more than 15KB, the twin-ballot is 6.5KB, while the ballot is 4.6KB. Which is quite reasonable in size.

Computations. We have implemented the voting process with both FHS and SDH signatures (centralized setup, twin-ballot generation, randomization and ballot-extraction, together with verification) in Rust, using `https://github.com/zkcrypto/bls12_381` library, with the two additional enhancements for Strong Receipt-Freeness, but without any implementation optimizations. On Figure 1, we deal with the 1-out-of-25 ballot case. The most important for practical use is the quite efficient generation of the full twin-ballots, with cross-compiled applications on mobile phones (see Figure 2, for

	param		Twin-ballot		Ballot	
	Generate	Verify	Generate	Verify	Rand-Extract	Verify
SDH	1225ms	4062ms	179ms	390ms	66ms	211ms
FHS	1019ms	3710ms	142ms	339ms	49ms	241ms

Fig. 1. Implementation of 1-out-of-25 Ballots (Rust on Macbook Pro 2021 – M1 Pro)

the 1-out-of-n case). Encryption, signing key generation, and signature are all linear in n. The proof of validity (for one constraint) is almost constant. If we approximate the time as an affine function in $a \times n + b$, the time for each proof of subset membership is thus b: not much more than 100ms on a 6-year-old smartphone.

	$n = 5$	$n = 10$	$n = 25$	a	b
Using Linearly-Homomorphic Proofs, cross-compiled from Rust					
2017 – SnapDragon 835 (Android)	243ms	375ms	710ms	23.1ms	134.5ms
2019 – Intel i9-9880H (MacOS)	91ms	134ms	261ms	8.5ms	48.8ms
2021 – iPhone 13 A15 Bionic (iOS)	43ms	63ms	125ms	4.1ms	22.2ms

Fig. 2. Voting time (encrypt, prove, generate keys, and sign) for 1-out-of-n Choices

Privacy of the Votes. The first important property in electronic voting is the privacy of the vote. The voter derives the twin-ciphertexts $(C_0, \vec{C}) = (r \cdot P, \vec{M}_j + r \cdot \vec{Z})$ and $(D_0, \vec{D}) = (s \cdot P, s \cdot \vec{Z})$, the twin-tags $\vec{\tau}'_0 = (r+1) \cdot \vec{\tau}_j$, $\vec{\tau}'_1 = s \cdot \vec{\tau}_j$, and randomized proofs $\mathsf{proof}'_0, \mathsf{proof}'_1, \mathsf{com}'$, for random $r, s, \nu' \xleftarrow{\$} \mathbb{Z}_p$, which are unlinkable to $\vec{M}_j$.

In order to show it, let us denote $(\vec{\tau} = (P, U = x \cdot P, V = x^2 \cdot P), \mathsf{proof} = (\Theta = \nu \cdot P, \Psi = \nu \cdot U), \mathsf{com} = (\hat{C} = x \cdot \hat{V}_{2,1} + \nu \cdot \hat{V}_{1,1}, \hat{D} = x \cdot \hat{V}_{2,2} + \nu \cdot \hat{V}_{1,2}))$ the proven SDH tag for $\vec{M}_j$. Using random $r, s, \nu' \xleftarrow{\$} \mathbb{Z}_p$, the voter generates

$$\begin{aligned} (C_0, \vec{C}) &= r \cdot (P, \vec{Z}) + (0, \vec{M}_j) & (D_0, \vec{D}) &= s \cdot (P, \vec{Z}) \\ \vec{\tau}'_0 &= (r+1) \cdot (P, U, V) & \vec{\tau}'_1 &= s \cdot (P, U, V) \\ \mathsf{proof}'_0 &= (r+1) \cdot [(\Theta, \Psi) + \nu' \cdot (P, U)] & \mathsf{proof}'_1 &= s \cdot [(\Theta, \Psi) + \nu' \cdot (P, U)] \end{aligned}$$

$$\mathsf{com}' = (\hat{C}, \hat{D}) + \nu' \cdot (\hat{V}_{1,1}, \hat{V}_{1,2}) = x \cdot (\hat{V}_{2,1}, \hat{V}_{2,2}) + (\nu + \nu') \cdot (\hat{V}_{1,1}, \hat{V}_{1,2})$$

One can indeed note that

$$\begin{aligned} \mathsf{proof}'_0 &= (r+1) \cdot [(\Theta, \Psi) + \nu' \cdot (P, U)] & \mathsf{proof}'_1 &= s \cdot [(\Theta, \Psi) + \nu' \cdot (P, U)] \\ &= (r+1) \cdot [\nu \cdot (P, U) + \nu' \cdot (P, U)] & &= s \cdot [\nu \cdot (P, U) + \nu' \cdot (P, U)] \\ &= (\nu + \nu') \cdot [(r+1) \cdot (P, U)] & &= (\nu + \nu') \cdot [s \cdot (P, U)] \end{aligned}$$

are valid proofs for $\vec{\tau}'_0$ and $\vec{\tau}'_1$ with com'.

Let us replace the CRS $(\hat{V}_{1,1}, \hat{V}_{1,2}, \hat{V}_{2,1}, \hat{V}_{2,2})$ with a Diffie-Hellman tuple $(\hat{V}_{1,1}, \hat{V}_{1,2}, \hat{V}_{2,1} = \alpha \cdot \hat{V}_{1,1}, \hat{V}_{2,2} = \alpha \cdot \hat{V}_{1,2})$. This is indistinguishable under the DDH assumption in $\hat{\mathbb{G}}$. Then $\mathsf{com}' = (\alpha x + (\nu + \nu')) \cdot (\hat{V}_{1,1}, \hat{V}_{1,2})$.

If we define $\mathsf{com}' = \beta \cdot (\hat{V}_{1,1}, \hat{V}_{1,2})$, for a random $\beta \xleftarrow{\$} \mathbb{Z}_p$, which implicitly and randomly defines ν', and if we denote, for random $r, s \xleftarrow{\$} \mathbb{Z}_p$,

$$(P_0, U_0, V_0, \vec{Z}_0) = (r+1) \cdot (P, U, V, \vec{Z}) \qquad (P_1, U_1, V_1, \vec{Z}_1) = s \cdot (P, U, V, \vec{Z})$$

$$\begin{aligned} (C_0, \vec{C}) &= (P_0 - P, \vec{Z}_0 - \vec{Z}) + (0, \vec{M}_j) & (D_0, \vec{D}) &= (P_1, \vec{Z}_1) \\ \vec{\tau}'_0 &= (P_0, U_0, V_0) & \vec{\tau}'_1 &= (P_1, U_1, V_1) \\ \mathsf{proof}'_0 &= (\nu + \nu') \cdot (P_0, U_0) & \mathsf{proof}'_1 &= (\nu + \nu') \cdot (P_1, U_1) \\ &= (\beta - \alpha x) \cdot (P_0, U_0) & &= (\beta - \alpha x) \cdot (P_1, U_1) \\ &= \beta \cdot (P_0, U_0) - \alpha \cdot (U_0, V_0) & &= \beta \cdot (P_1, U_1) - \alpha \cdot (U_1, V_1) \end{aligned}$$

Let us replace $(P_0, U_0, V_0, \vec{Z}_0)$ and $(P_1, U_1, V_1, \vec{Z}_1)$ by random tuples, which is indistinguishable under the DDH assumption in $\mathbb{G}$. This is also under the organizational assumption that the scalars in the tags generated during the setup are really private. As $\vec{Z}_0$ is random, we can note $(C_0, \vec{C}) = (P_0 - P, \vec{Z}_0)$.

For a new random $y \overset{\$}{\leftarrow} \mathbb{Z}_p$, we can go back to $(P_0, U_0, V_0) = (r+1) \cdot (P, y \cdot P, y^2 \cdot P)$ and $(P_1, U_1, V_1, \vec{Z}_1) = s \cdot (P, y \cdot P, y^2 \cdot P, \vec{Z})$, which is indistinguishable under the DDH and the SDH assumptions in $\mathbb{G}$. This then allows to honestly generate the proofs using y, even for a valid CRS $(\hat{V}_{1,1}, \hat{V}_{1,2}, \hat{V}_{2,1}, \hat{V}_{2,2})$ that is no longer a Diffie-Hellman tuple. The proven twin-tags are thus unlinkable to $\vec{\tau}_j$, and the ciphertext $(C_0, \vec{C})$ encrypts a random plaintext. This is perfectly private.

We can stress that the ballot privacy (in the above honest-but-curious setting) is achieved in the standard model, under the SXDH assumption and the SDH assumption in $\mathbb{G}$.

Strong Receipt-Freeness. But we need stronger privacy properties, by preventing malicious voter behaviors, with replay and vote-selling attacks. When the voter has sent the ciphertext $(C_0, \vec{C})$, he can open it with his randomness r. But the ciphertext in the public ballot-box is $(C'_0, \vec{C'})$ that has been randomized with an unknown r': this excludes VS-attacks. We stress that the receiver must first check $D_0 \neq 0$ for randomization to be effective. The additional Diffie-Hellman proof for $(P, T = \mathcal{H}(\mathsf{vk}), C'_0, W)$ avoids replay attacks under a different signing key vk'. Excluding multiple ballots with the same vk then avoids replay attacks, and thus CS-attacks. But here, both GGM and ROM are needed.

To assess strong receipt-freeness, we can follow a similar path as in [CCFG16, Theorem 3 – Figure 1], with their security game for the strong receipt-freeness (a stronger privacy notion, against malicious voters). The proof is performed in two mains steps, with first the RCCA-security for our encryption scheme, thanks to the additional Diffie-Hellman proof, and then the privacy of the ciphertexts. But we cannot use the same approach as they do, for simulating the decryption oracle, as we take advantage of the compact randomness-reuse version of ElGamal. However, we can exploit the GGM/AGM extractor: any adversarially-generated Diffie-Hellman tuple $(P, \mathcal{H}(\mathsf{vk}), C_0, W_0)$, for a *truly random* $T = \mathcal{H}(\mathsf{vk})$, as we are in the ROM, allows to extract r such that $C_0 = r \cdot P$. This thus allows to simulate the decryption of any ballot with a valid signature under a *fresh* vk. Let us now simulate the public view of an adversary against the strong receipt-freeness:

1. in the initial security game, honest published ballots are generated as fresh ciphertexts, and all the adversarial ballots are honestly randomized before publication. Final tally decryption uses the decryption key and the Schnorr proof is performed honestly;
2. the Schnorr proof of correct tally decryption is simulated (by programming the random oracle used to derive the challenge);
3. the adversarial Groth-Sahai proofs of Diffie-Hellman tuples are additionally checked knowing the discrete logarithm of $\mathcal{H}$ values (by programming the random oracle);
4. the adversarial twin-ballots, with *fresh* vk, are decrypted using the above RCCA-security: valid fresh Diffie-Hellman tuples $(P, T = \mathcal{H}(\mathsf{vk}), C_0, W_0)$ and $(P, T = \mathcal{H}(\mathsf{vk}), D_0, W_1)$ come with r and s, such that $C_0 = r \cdot P$ and $D_0 = s \cdot P$, respectively, which allows to decrypt, using the GGM/AGM extractor. A ballot with a replayed vk or invalid W_0, W_1 is rejected. The tally is computed on clear data, which does not impact the proof of correct decryption, that was already simulated;
5. from r and s, in the adversarial twin-ballots with *fresh* vk, one can recompute the tags $\vec{\tau}'_0$ and $\vec{\tau}'_1$. If these tags are not the expected ones, the ballots are rejected;
6. the simulator learns the users' signing keys sk associated to each sent vk, using the GGM/AGM extractor for adversarially generated vk, and keeps the signing keys SK generated during the setup;
7. for published ballots, the Groth-Sahai proofs of SDH tuples are simulated using a Diffie-Hellman CRS $(\hat{V}_{1,1}, \hat{V}_{1,2}, \hat{V}_{2,1}, \hat{V}_{2,2})$ and signatures are generated using the signing keys;
8. for published ballots, fresh random ciphertexts and random tags are generated (proofs are simulated and signatures use the signing keys).

We have built a simulator that generates views indistinguishable from real executions, without leaking any information about the votes. We stress that this stronger privacy in the malicious setting relies on the GGM and the random oracle model. They are widely admitted, in particular for efficient constructions.

Strong Verifiability. From the unforgeability under chosen-message attacks of the $\mathsf{OT\text{-}LH\text{-}Sign}$ in the GGM, and the non-miscibility of the SDH-tags (Proposition 2), the validity of signatures on $(P_{\mathcal{S}}, C_0, \vec{C})$, $(0, D_0, \vec{D})$ or $(P_{\mathcal{S}}, C'_0, \vec{C}')$ under the key VK (and a valid tag) implies that they are linear combinations of some $(P_{\mathcal{S}}, 0, \vec{M}_j)$ and $(0, P, \vec{Z})$, again under the assumption that the scalars in the tags generated during the setup are really private, and even destroyed after use. Hence, no new signature can be generated by anybody, excepted under linear combinations on equivalent tags. Because of the first component $P_{\mathcal{S}}$ (specific to $\mathcal{S}$) in $(P_{\mathcal{S}}, C'_0, \vec{C}')$, this is necessarily for a valid ballot, but not necessarily for the same j as in $(P_{\mathcal{S}}, C_0, \vec{C})$: $C_0 = r \cdot P$ and $\vec{C} = r \cdot \vec{Z} + \vec{M}_j$, and $C'_0 = r' \cdot P$ and $\vec{C}' = r' \cdot \vec{Z} + \vec{M}_k$. Anyway, they are both valid ballots. Furthermore, the decryption of the tally is given with a proof of correct decryption, which provides the **universal verifiability** of ballots and tally.

However, the voter needs the additional guarantee of no modification of his initial choice $\vec{M}_j$ in $(C_0, \vec{C})$ after randomization in $(C'_0, \vec{C}')$, by the receiver: the voter, who has kept $T = \mathcal{H}(\mathsf{vk})$, can ask for his vote, and check the validity of σ' with respect to $(P, C'_0, \vec{C}')$ under vk. This implies $(C'_0, \vec{C}')$ can only be a randomization of the initial ciphertext $(C_0, \vec{C})$: $\vec{M}_k = \vec{M}_j$, under the unforgeability of the $\mathsf{OT\text{-}LH\text{-}Sign}$, as vk is specific to a voter and used only once. This **individual verifiability** convinces every voter of the integrity of the ballot-box, with the presence of his vote.

References

BCC+16. Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 327–357. Springer, Heidelberg, May 2016.

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

BFI+10. Olivier Blazy, Georg Fuchsbauer, Malika Izabachène, Amandine Jambert, Hervé Sibert, and Damien Vergnaud. Batch Groth-Sahai. In Jianying Zhou and Moti Yung, editors, *ACNS 10*, volume 6123 of *LNCS*, pages 218–235. Springer, Heidelberg, June 2010.

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.

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

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

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

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

DPP22. Henri Devillez, Olivier Pereira, and Thomas Peters. Traceable receipt-free encryption. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part III*, volume 13793 of *LNCS*, pages 273–303. Springer, Heidelberg, December 2022.

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

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

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

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

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.

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

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

JR13. Charanjit S. Jutla and Arnab Roy. Shorter quasi-adaptive NIZK proofs for linear subspaces. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 1–20. Springer, Heidelberg, December 2013.

KW15. Eike Kiltz and Hoeteck Wee. Quasi-adaptive NIZK for linear subspaces revisited. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 101–128. Springer, Heidelberg, April 2015.

LCKO19. Jiwon Lee, Jaekyoung Choi, Jihye Kim, and Hyunok Oh. SAVER: Snark-friendly, additively-homomorphic, and verifiable encryption and decryption with rerandomization. Cryptology ePrint Archive, Report 2019/1270, 2019. https://eprint.iacr.org/2019/1270.

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.

MMR22. D. Mestel, J. Muller, and P. Reisert. How Efficient are Replay Attacks against Vote Privacy? A Formal Quantitative Analysis. In Stefano Calzavara and David Naumann, editors, *CSF 2022 Computer Security Foundations Symposium*, pages 179–194. IEEE Computer Society, 2022.

PHGR13. Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE Computer Society Press, May 2013.

Wat05. Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127. Springer, Heidelberg, May 2005.

A Complete Protocol for Proof of Subset Membership on ElGamal Ciphertexts

In this appendix, we describe how to use Linearly-Homomorphic Signatures with Randomizable Tags (LH-Sign-RTag) for zero-knowledge proofs of subset membership on ElGamal ciphertexts. We propose additional, but optional, steps to make them non-malleable and traceable by the prover. Then, combinations of such proofs can be used for any kind of election, in order to prove the validity of the ballot, with randomizability to provide receipt-freeness (to exclude VS-attacks), and non-malleability to provide the strong receipt-freeness (to exclude CS-attacks). The process involves

- an authority (possibly distributed among several parties), that generates the global parameters;
- a sender, that generates the ciphertext with a (non-malleable) proof of subset membership;
- a receiver, that (randomizes and) stores the final ciphertext.

We will use a pairing-friendly setting $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, p, P, \hat{P}, e)$, where the SXDH assumption holds, for the ElGamal encryption privacy.

We thus consider the set $\mathcal{S} = \{\vec{x}_1, \dots, \vec{x}_N\} \subseteq \mathbb{Z}_p^n$ of the N authorized plaintexts, and denote $P_\mathcal{S} \in \mathbb{G}$, an element that characterizes the proof on $\mathcal{S}$ (in case of multiple concurrent proofs, on the same ciphertext). It can be $P_\mathcal{S} = \mathcal{H}(\mathcal{S})$, where $\mathcal{H}$ is a full-domain hash function into $\mathbb{G}$.

A.1 Generic Approach

We first present the generic approach, with an LH-Sign-RTag used in black box. This illustrates the main steps, that are required or optional, according to the needs. We ignore here some classical steps for Groth-Sahai proofs, of Diffie-Hellman tuples.

Initialisation. The authority generates

Encryption Keys: A pair of ElGamal encryption keys $(\mathsf{DK}, \mathsf{EK})$ for plaintexts in $\mathbb{G}^n$: the private decryption key is $\mathsf{DK} = \vec{z} \stackrel{\$}{\leftarrow} \mathbb{Z}_p^n$, and the public encryption key is $\mathsf{EK} = \vec{Z} = \vec{z} \cdot P \in \mathbb{G}^n$;

Signature Keys: A pair of LH-Sign-RTag signing keys $(\mathsf{SK}, \mathsf{VK})$ for messages in $\mathbb{G}^{n+2}$;

Verifiable Tags: N verifiable tags Tag_j, for $j = 1, \dots, N$;

Initial Signatures: N pairs of signatures $(\Sigma_{j,0}, \Sigma_{j,1})$, for $j = 1, \dots, N$, on $\vec{M}_j = \vec{x}_j \cdot P \in \mathbb{G}^n$, with two additional components:

$$\Sigma_{j,0} = \mathsf{Sign}(\mathsf{SK}, \mathsf{Tag}_j, (P_\mathcal{S}, 0, \vec{M}_j)) \qquad \Sigma_{j,1} = \mathsf{Sign}(\mathsf{SK}, \mathsf{Tag}_j, (0, P, \vec{Z}))$$

The public parameters thus consist of EK, VK, and $(\mathsf{Tag}_j, \vec{x}_j, \Sigma_{j,0}, \Sigma_{j,1})$ for $j = 1, \ldots, N$. The signing key SK and the random values must be forgotten, except the decryption key DK, that must be kept for the final tally.

Ciphertext with Proof of Subset Membership. From the above public elements, the sender generates

***Ciphertext*:** The ElGamal ciphertext $(C_0, \vec{C})$ for his authorized plaintext $\vec{x} = \vec{x}_j \in \mathcal{S} \subseteq \mathbb{Z}_p^n$, which determines the by-now fixed index j, with a random scalar $r \xleftarrow{\$} \mathbb{Z}_p$:

$$C_0 = r \cdot P \qquad \vec{C} = r \cdot \vec{Z} + \vec{M} \qquad \text{with } \vec{M} = \vec{M}_j = \vec{x}_j \cdot P \in \mathbb{G}^n$$

***Proof of Subset Membership*:** The proof, from Tag_j and $\Sigma_{j,0}, \Sigma_{j,1}$:

$$\Sigma_0 = \Sigma_{j,0} + r \cdot \Sigma_{j,1} \qquad \Sigma_1 = \Sigma_{j,1} \qquad \text{(optional for further randomization)}$$

which are valid signatures of $(P_{\mathcal{S}}, C_0, \vec{C})$ and $(0, P, \vec{Z})$, respectively, under VK for the unlinkable tag $\mathsf{Tag} = \mathsf{Tag}_j$. Granted the property of $\mathsf{LH\text{-}Sign\text{-}RTag}$, with randomizable tags, one can randomize Tag into Tag' and adapt Σ_0 and Σ_1 into Σ_0' and Σ_1', respectively for this new tag: Σ_0' and Σ_1' are valid signatures of $(P_{\mathcal{S}}, C_0, \vec{C})$ and $(0, P, \vec{Z})$, respectively, under VK for the tag Tag'.

***Individual Verifiability* (optional for further randomization):** A pair of $\mathsf{OT\text{-}LH\text{-}Sign}$ signing keys $(\mathsf{sk}, \mathsf{vk})$ for messages in $\mathbb{G}^{n+2}$, and the signatures

$$\sigma_0 = \mathsf{Sign}(\mathsf{sk}, (P, C_0, \vec{C})) \qquad \sigma_1 = \mathsf{Sign}(\mathsf{sk}, (0, P, \vec{Z}))$$

***Non-Malleability* (optional):** The Diffie-Hellman value $W = r \cdot T$, for $T = \mathcal{H}(\mathsf{vk}) \in \mathbb{G}$, together with a Groth-Sahai proof π of Diffie-Hellman tuple for (P, T, C_0, W).

The sender then outputs the ciphertext $(C_0, \vec{C})$, together with the proof that consists of $(\mathsf{Tag}', \Sigma_0', \Sigma_1')$, and (optionally) the signatures $(\mathsf{vk}, \sigma_0, \sigma_1)$ and (W, π).

Randomization. Before storing the ciphertext, the receiver may (optionally) randomize it, after having verified all the values $(C_0, \vec{C})$, $(\mathsf{Tag}', \Sigma_0', \Sigma_1')$, $(\mathsf{vk}, \sigma_0, \sigma_1)$, and (W, π):

***Verification*:** The receiver verifies
- vk is fresh (no repetition)
- the validity of Tag'
- the validity of the signatures Σ_0' and Σ_1' on $(P_{\mathcal{S}}, C_0, \vec{C})$ and $(0, P, \vec{Z})$, respectively, under VK for the tag Tag';
- the validity of the signatures σ_0 and σ_1 on $(P, C_0, \vec{C})$ and $(0, P, \vec{Z})$, respectively, under vk;
- the validity of the Diffie-Hellman proof π on the tuple $(P, T = \mathcal{H}(\mathsf{vk}), C_0, W)$.

***Ciphertext*:** The receiver randomizes the ElGamal ciphertext $(C_0, \vec{C})$ using a random scalar $s \xleftarrow{\$} \mathbb{Z}_p$

$$C_0' = C_0 + s \cdot P \qquad \vec{C}' = \vec{C} + s \cdot \vec{Z}$$

***Signatures*:** The receiver adapts the signatures for this new ciphertext

$$\Sigma' = \Sigma_0' + s \cdot \Sigma_1' \qquad \sigma = \sigma_0 + s \cdot \sigma_1$$

Then, Σ' is a valid signature of $(P_{\mathcal{S}}, C_0', \vec{C}')$ under VK for the tag Tag', and σ is a valid signature of $(P, C_0', \vec{C}')$ under vk. Granted the property of $\mathsf{LH\text{-}Sign\text{-}RTag}$, with randomizable tags, one can randomize Tag' into Tag'' and adapt Σ' into Σ'' for this new tag: Σ'' is a valid signature of $(P_{\mathcal{S}}, C_0', \vec{C}')$ under VK for the tag Tag''.

***Diffie-Hellman Proof*:** The receiver adapts the Groth-Sahai proof π of the Diffie-Hellman tuple $(P, T = \mathcal{H}(\mathsf{vk}), C_0, W)$, into a randomized proof π' of the Diffie-Hellman tuple $(P, T = \mathcal{H}(\mathsf{vk}), C_0', W')$, using s, for $W' = W + s \cdot T$.

The receiver then stores the ciphertext $(C_0', \vec{C}')$, together with the proof that consists of $(\mathsf{Tag}'', \Sigma'')$, the signature (vk, σ), and (W', π'), which can be publicly verified:

- Tag'' is a valid tag;
- Σ' is a valid signature on $(P_{\mathcal{S}}, C'_0, \vec{C}')$, under VK for the tag Tag''. This proves the plaintext must be in $\mathcal{S}$;
- σ is a valid signature on $(P, C'_0, \vec{C}')$ under vk. This ensures the sender that $(C'_0, \vec{C}')$ contains the same plaintext as his initial ciphertext $(C_0, \vec{C})$, but without knowing the actual randomness, which excludes VS-attacks;
- π' is a valid Diffie-Hellman proof on the tuple $(P, T = \mathcal{H}(\mathsf{vk}), C'_0, W')$. This ensures that only the owner of vk can have initiated the ciphertext, which excludes malleability or repetition, and thus CS-attacks.

Some Details. First, for the $\mathsf{OT\text{-}LH\text{-}Sign}$, one can use $\mathsf{sk} = \vec{s} \overset{\$}{\leftarrow} \mathbb{Z}_p^n$ and $\mathsf{vk} = \vec{s} \cdot \hat{P}$. Then, $\sigma = \mathsf{Sign}(\mathsf{sk}, \vec{M}) = \langle \vec{s}, \vec{M} \rangle = \sum s_i \cdot M_i \in \mathbb{G}$ can be verified as $e(\sigma, \hat{P}) = \prod e(M_i, \mathsf{vk}_i)$. Below, we instantiate $\mathsf{LH\text{-}Sign\text{-}RTag}$ from this $\mathsf{OT\text{-}LH\text{-}Sign}$, with two different kinds of verifiable tags.

For Groth-Sahai proofs, we need a random tuple $(\hat{V}_{1,1}, \hat{V}_{1,2}, \hat{V}_{2,1}, \hat{V}_{2,2}) \in \hat{\mathbb{G}}^4$, and the proof $\pi = (\mathsf{com}, \mathsf{proof})$, consists of a commitment com of the secret witness known by the prover, and a proof proof, both being additively homomorphic, and publicly verifiable. We will not detail the properties and the computations for those proofs, as they are classical. But we stress they can be randomized.

A.2 FHS Instantiation

The FHS $\mathsf{LH\text{-}Sign\text{-}RTag}$ (which stands for Fuchsbauer-Hanser-Slamanig [FHS19]) uses tags of the form $\mathsf{Tag} = (\tau_1 = 1/t \cdot P, \tau_2 = 1/t \cdot \hat{P}) \in \mathbb{G} \times \hat{\mathbb{G}}$, for a scalar $t \overset{\$}{\leftarrow} \mathbb{Z}_p$. It is self-verifiable as one can check that $e(P, \tau_2) = e(\tau_1, \hat{P})$. The signature of $\vec{M} = (M_i) \in \mathbb{G}^n$ is $\Sigma = t \cdot (\sum s_i \cdot M_i) \in \mathbb{G}$, under $\mathsf{VK} = \vec{s} \cdot \hat{P} \in \hat{\mathbb{G}}^n$, that can be verified by $e(\Sigma, \tau_2) = \prod e(M_i, \mathsf{VK}_i)$.

One can easily randomize Tag and adapt the signature Σ, in a perfectly unlinkable way: $\mathsf{Tag}' = 1/t' \cdot \mathsf{Tag} = (\tau'_1 = 1/t' \cdot \tau_1, \tau'_2 = 1/t' \cdot \tau_2)$, for a scalar $t' \overset{\$}{\leftarrow} \mathbb{Z}_p$, and $\Sigma' = t' \cdot \Sigma$.

Granted the self-verifiable tags, this approach is quite efficient, from both the computation and communication points of view. However, the initialisation cannot efficiently be distributed among multiple users, to avoid a unique trusted party. The verifiable tags and the initial signatures must be generated on a secure device. But the process is quite close to the above generic one.

Initialisation. The authority generates

***Encryption Keys*:** A pair of ElGamal encryption keys $(\mathsf{DK}, \mathsf{EK})$ for plaintexts in $\mathbb{G}^n$: the private decryption key is $\mathsf{DK} = \vec{z} \overset{\$}{\leftarrow} \mathbb{Z}_p^n$, and the public encryption key is $\mathsf{EK} = \vec{Z} = \vec{z} \cdot P \in \mathbb{G}^n$;

***Signature Keys*:** A pair of FHS $\mathsf{LH\text{-}Sign\text{-}RTag}$ signing keys $\mathsf{SK} = \vec{s} \overset{\$}{\leftarrow} \mathbb{Z}_p^{n+2}$, and $\mathsf{VK} = \vec{s} \cdot \hat{P} \in \hat{\mathbb{G}}^{n+2}$;

***Verifiable Tags*:** N verifiable tags $\mathsf{Tag}_j = (\tau_{j,1} = 1/t_j \cdot P, \tau_{j,2} = 1/t_j \cdot \hat{P})$, with $t_j \overset{\$}{\leftarrow} \mathbb{Z}_p$, for $j = 1, \ldots, N$;

***Initial Signatures*:** N pairs of signatures, for $j = 1, \ldots, N$, on $\vec{M}_j = \vec{x}_j \cdot P = (M_{j,1}, \ldots, M_{j,n}) \in \mathbb{G}^n$:

$$\Sigma_{j,0} = t_j \cdot (s_1 \cdot P_{\mathcal{S}} + \sum_{i=1}^{n} s_{i+2} \cdot M_{j,i}) \qquad \Sigma_{j,1} = t_j \cdot (s_2 \cdot P + \sum_{i=1}^{n} s_{i+2} \cdot Z_i)$$

The public parameters thus consist of EK, VK, and $(\mathsf{Tag}_j, \vec{x}_j, \Sigma_{j,0}, \Sigma_{j,1})$ for $j = 1, \ldots, N$. The signing key SK and the random values must be forgotten, except the decryption key DK.

Ciphertext with Proof of Subset Membership. From the above public elements, the sender generates

***Ciphertext*:** The ElGamal ciphertext $(C_0, \vec{C})$ for his authorized plaintext $\vec{x} = \vec{x}_j \in \mathcal{S} \subseteq \mathbb{Z}_p^n$, which determines the by-now fixed index j, with a random scalar $r \overset{\$}{\leftarrow} \mathbb{Z}_p$:

$$C_0 = r \cdot P \qquad \vec{C} = r \cdot \vec{Z} + \vec{M} \qquad \text{with } \vec{M} = \vec{M}_j = \vec{x}_j \cdot P \in \mathbb{G}^n$$

***Proof of Subset Membership*:** The proof, from Tag_j and $\Sigma_{j,0}, \Sigma_{j,1}$:

$$\Sigma_0 = \Sigma_{j,0} + r \cdot \Sigma_{j,1} \qquad\qquad \Sigma_1 = \Sigma_{j,1}$$

which are valid signatures of $(P_{\mathcal{S}}, C_0, \vec{C})$ and $(0, P, \vec{Z})$, respectively, under VK for the tag $\mathsf{Tag} = \mathsf{Tag}_j$. Granted the property of $\mathsf{LH\text{-}Sign\text{-}RTag}$, with randomizable tags, one can randomize $\mathsf{Tag} = (\tau_1, \tau_2)$ into $\mathsf{Tag}' = (\tau_1', \tau_2')$, for a random $t' \xleftarrow{\$} \mathbb{Z}_p$, and adapt Σ_0 and Σ_1 into Σ_0' and Σ_1', respectively for this new tag:

$$\mathsf{Tag}' = (\tau_1' = 1/t' \cdot \tau_1, \tau_2' = 1/t' \cdot \tau_2) \qquad \Sigma_0' = t' \cdot \Sigma_0 \qquad \Sigma_1' = t' \cdot \Sigma_1$$

***Individual Verifiability*:** A pair of $\mathsf{OT\text{-}LH\text{-}Sign}$ signing keys $\mathsf{sk} = (\mathsf{sk}_i)_i \xleftarrow{\$} \mathbb{Z}_p^{n+2}$, and $\mathsf{vk} = \mathsf{sk} \cdot \hat{P} = (\mathsf{vk}_i)_i \in \hat{\mathbb{G}}^{n+2}$, and the signatures

$$\sigma_0 = \mathsf{sk}_1 \cdot P + \mathsf{sk}_2 \cdot C_0 + \sum_{i=1}^{n} \mathsf{sk}_{i+2} \cdot C_i \qquad \sigma_1 = \mathsf{sk}_2 \cdot P + \sum_{i=1}^{n} \mathsf{sk}_{i+2} \cdot Z_i$$

***Non-Malleability*:** The Diffie-Hellman value $W = r \cdot T$, for $T = \mathcal{H}(\mathsf{vk}) \in \mathbb{G}$, together with a Groth-Sahai proof $\pi = (\mathsf{com}, \mathsf{proof})$ of Diffie-Hellman tuple for (P, T, C_0, W).

The sender then outputs the ciphertext $(C_0, \vec{C})$, together with the proof that consists of $(\mathsf{Tag}', \Sigma_0', \Sigma_1')$, the signatures $(\mathsf{vk}, \sigma_0, \sigma_1)$, and (W, π).

Randomization. Before storing the ciphertext, the receiver randomizes it, after having verified all the values $(C_0, \vec{C})$, $(\mathsf{Tag}', \Sigma_0', \Sigma_1')$, $(\mathsf{vk}, \sigma_0, \sigma_1)$, and (W, π):

***Verification*:** The receiver verifies

vk is fresh (no repetition)

The tag $\mathsf{Tag}' = (\tau_1', \tau_2')$ $\qquad e(P, \tau_2') = e(\tau_1', \hat{P})$

The signature Σ_0' $\qquad e(\Sigma_0', \tau_2') = e(P_{\mathcal{S}}, \mathsf{VK}_1) \cdot e(C_0, \mathsf{VK}_2) \cdot \prod_{i=1}^{n} e(C_i, \mathsf{VK}_{i+2})$

The signature Σ_1' $\qquad e(\Sigma_1', \tau_2') = e(P, \mathsf{VK}_2) \cdot \prod_{i=1}^{n} e(Z_i, \mathsf{VK}_{i+2})$

The signature σ_0 $\qquad e(\sigma_0, \hat{P}) = e(P, \mathsf{vk}_1) \cdot e(C_0, \mathsf{vk}_2) \cdot \prod_{i=1}^{n} e(C_i, \mathsf{vk}_{i+2})$

The signature σ_1 $\qquad e(\sigma_1, \hat{P}) = e(P, \mathsf{vk}_2) \cdot \prod_{i=1}^{n} e(Z_i, \mathsf{vk}_{i+2})$

and the validity of the Diffie-Hellman proof $\pi = (\mathsf{com}, \mathsf{proof})$ on the tuple $(P, T = \mathcal{H}(\mathsf{vk}), C_0, W)$.

***Ciphertext*:** The receiver randomizes the ElGamal ciphertext $(C_0, \vec{C})$ using a random scalar $s \xleftarrow{\$} \mathbb{Z}_p$

$$C_0' = C_0 + s \cdot P \qquad\qquad \vec{C}' = \vec{C} + s \cdot \vec{Z}$$

***Signatures*:** The receiver randomizes the tag, with $t'' \xleftarrow{\$} \mathbb{Z}_p$, and adapts the signatures to the new ciphertext and the new tag

$$\mathsf{Tag}'' = (\tau_1'' = 1/t'' \cdot \tau_1', \tau_2'' = 1/t'' \cdot \tau_2') \qquad \Sigma'' = t'' \cdot (\Sigma_0' + s \cdot \Sigma_1') \qquad \sigma = \sigma_0 + s \cdot \sigma_1$$

***Diffie-Hellman Proof*:** The receiver adapts the Groth-Sahai proof π of the Diffie-Hellman tuple $(P, T = \mathcal{H}(\mathsf{vk}), C_0, W)$, into a randomized proof π' of the Diffie-Hellman tuple $(P, T = \mathcal{H}(\mathsf{vk}), C_0', W')$, using s, for $W' = W + s \cdot T$.

The receiver then stores the ciphertext $(C_0', \vec{C}')$, together with the proof that consists of $(\mathsf{Tag}'', \Sigma'')$, the signature (vk, σ), and (W', π'), which can be publicly verified.

B SDH Instantiation

The SDH $\mathsf{LH\text{-}Sign\text{-}RTag}$ (which stands for Square Diffie-Hellman) uses tags of the form $(\tau_1 = U, \tau_2 = t \cdot U, \tau_3 = t^2 \cdot U)$, for $U \in \mathbb{Z}_p$ and $t \xleftarrow{\$} \mathbb{Z}_p$, which require additional validity proofs, in order to be verifiable. This can be done with a Groth-Sahai proof of Diffie-Hellman tuple for $(\tau_1, \tau_2, \tau_2, \tau_3)$, with a commitment $\mathsf{com} \in \hat{\mathbb{G}}^2$ and a pair of proofs $\mathsf{proof} \in \mathbb{G}^2$. Hence, a verifiable SDH tag is $\mathsf{Tag} = (\tau_1, \tau_2, \tau_3, \pi = (\mathsf{com}, \mathsf{proof})) \in \mathbb{G}^5 \times \hat{\mathbb{G}}^2$.

The signature of $\vec{M} = (M_i) \in \mathbb{G}^n$ is $\Sigma = (\sum s_i \cdot M_i) + s_{n+1} \cdot \tau_1 + s_{n+2} \cdot \tau_2 + s_{n+3} \cdot \tau_3 \in \mathbb{G}$, under $\mathsf{VK} = (\mathsf{VK}_i = s_i \cdot \hat{P})_{i=1}^{n+3}$, that can be verified by $e(\Sigma, \hat{P}) = \prod_{i=1}^{n} e(M_i, \mathsf{VK}_i) \times \prod_{i=1}^{3} e(\tau_i, \mathsf{VK}_{n+i})$. Actually, τ_1 will always have a particular form, we will be able to ignore it, reducing the dimension to $n + 2$.

However, when combining two signatures Σ_0 and Σ_1 on two messages $\vec{M}$ and $\vec{M}'$, with coefficient α and α' under the same (or equivalent) tags Tag and Tag', one gets a signature on $\alpha \cdot \vec{M} + \alpha' \cdot \vec{M}'$ under the equivalent tag $(\tau_1'', \tau_2'', \tau_3'') = \alpha \cdot (\tau_1, \tau_2, \tau_3) + \alpha' \cdot (\tau_1', \tau_2', \tau_3')$. The proof can be adapted and randomized (if they both have the same com).

Because of the need of additional proofs of validity for the tags, this approach is a bit less efficient than the previous one, while still constant-time for the generation of the proofs of subset membership (independently of the size of the subset). However, the initialisation can efficiently be distributed among multiple users, which avoids a unique trusted party. Furthermore, the randomisation can also be distributed among multiple players.

The global process is a bit different from the above generic one, because of the evolution of the tag in case of combinations. We are thus more exhaustive for this description, for a later security analysis.

B.1 Description

Initialisation. The authority generates

***Independent Generators*:** Four random independent generators $(\hat{V}_{1,1}, \hat{V}_{1,2}, \hat{V}_{2,1}, \hat{V}_{2,2}) \in \hat{\mathbb{G}}^4$, that is likely a non-Diffie-Hellman tuple, for the Groth-Sahai proofs;

***Encryption Keys*:** A pair of ElGamal encryption keys $(\mathsf{DK}, \mathsf{EK})$ for plaintexts in $\mathbb{G}^n$: the private decryption key is $\mathsf{DK} = \vec{z} \xleftarrow{\$} \mathbb{Z}_p^n$, and the public encryption key is $\mathsf{EK} = \vec{Z} = \vec{z} \cdot P \in \mathbb{G}^n$;

***Signature Keys*:** A pair of SDH $\mathsf{LH\text{-}Sign\text{-}RTag}$ signing keys $\mathsf{SK} = \vec{s} \xleftarrow{\$} \mathbb{Z}_p^{n+4}$, and $\mathsf{VK} = \vec{s} \cdot \hat{P} \in \hat{\mathbb{G}}^{n+4}$;

***Verifiable Tags*:** N verifiable tags $(\tau_{j,1} = P, \tau_{j,2} = t_j \cdot P, \tau_{j,3} = t_j^2 \cdot P)$, with $t_j \xleftarrow{\$} \mathbb{Z}_p$, for $j = 1, \ldots, N$, with their validity proofs $\pi_j = (\mathsf{com}_j, \mathsf{proof}_j)$:

- commitment of the witness t_j, $\mathsf{com}_j = (\hat{C}_j = t_j \cdot \hat{V}_{2,1} + \mu_j \cdot \hat{V}_{1,1}, \hat{D}_j = t_j \cdot \hat{V}_{2,2} + \mu_j \cdot \hat{V}_{1,2})$, for $\mu_j \xleftarrow{\$} \mathbb{Z}_p$,
- and the pair of proofs $\mathsf{proof} = (\Theta_j = \mu_j \cdot P, \Psi_j = \mu_j \cdot \tau_{j,2})$, which satisfy[1]

$$\begin{aligned} e(P, \hat{C}_j) &= e(\tau_{j,2}, \hat{V}_{2,1}) \cdot e(\Theta_j, \hat{V}_{1,1}) & \qquad e(P, \hat{D}_j) &= e(\tau_{j,2}, \hat{V}_{2,2}) \cdot e(\Theta_j, \hat{V}_{1,2}) \\ e(\tau_{j,2}, \hat{C}_j) &= e(\tau_{j,3}, \hat{V}_{2,1}) \cdot e(\Psi_j, \hat{V}_{1,1}) & \qquad e(\tau_{j,2}, \hat{D}_j) &= e(\tau_{j,3}, \hat{V}_{2,2}) \cdot e(\Psi_j, \hat{V}_{1,2}) \end{aligned}$$

The verifiable tags thus consist of $\mathsf{Tag}_j = (\tau_{j,2}, \tau_{j,3}, \mathsf{com}_j = (\hat{C}_j, \hat{D}_j), \mathsf{proof}_j = (\Theta_j, \Psi_j)) \in \mathbb{G}^4 \times \hat{\mathbb{G}}^2$, as $\tau_{j,1} = P$;

***Initial Signatures*:** N pairs of signatures, for $j = 1, \ldots, N$, on $(P_{\mathcal{S}}, 0, \vec{M}_j)$ for $\vec{M}_j = \vec{x}_j \cdot P$ and $(0, P, \vec{Z})$, respectively, under VK and tag Tag_j:

$$\Sigma_{j,0} = s_1 \cdot P_{\mathcal{S}} + \sum_{i=1}^{n} s_{i+2} \cdot M_{j,i} + s_{n+3} \cdot \tau_{j,2} + s_{n+4} \cdot \tau_{j,3}$$

$$\Sigma_{j,1} = s_2 \cdot P + \sum_{i=1}^{n} s_{i+2} \cdot Z_i + s_{n+3} \cdot \tau_{j,2} + s_{n+4} \cdot \tau_{j,3}$$

[1] We stress that the four equations can be checked all at once, with three pairings only [BFI+10].

which satisfy

$$e(\Sigma_{j,0},\hat{P}) = e(P_{\mathcal{S}},\mathsf{VK}_1)\cdot\prod_{i=1}^{n} e(M_{j,i},\mathsf{VK}_{i+2})\cdot e(\tau_{j,2},\mathsf{VK}_{n+3})\cdot e(\tau_{j,3},\mathsf{VK}_{n+4})$$
$$e(\Sigma_{j,1},\hat{P}) = e(P,\mathsf{VK}_2)\cdot\prod_{i=1}^{n} e(Z_i,\mathsf{VK}_{i+2})\cdot e(\tau_{j,2},\mathsf{VK}_{n+3})\cdot e(\tau_{j,3},\mathsf{VK}_{n+4})$$

The public parameters thus consist of $(\hat{V}_{i,j}), \mathsf{EK}, \mathsf{VK}$, and $(\mathsf{Tag}_j, \vec{x}_j, \Sigma_{j,0}, \Sigma_{j,1})$ for $j = 1,\ldots,N$. The signing key SK and the random values must be forgotten, except the decryption key DK.

Ciphertext with Proof of Subset Membership. From the above public elements, the sender generates

***Ciphertext*:** The ElGamal ciphertext $(C_0,\vec{C})$ for his authorized plaintext $\vec{x} = \vec{x}_j \in \mathcal{S} \subseteq \mathbb{Z}_p^n$, which determines the by-now fixed index j, with a random scalar $r \overset{\$}{\leftarrow} \mathbb{Z}_p$:

$$C_0 = r\cdot P \qquad \vec{C} = r\cdot\vec{Z}+\vec{M} \qquad \text{with } \vec{M} = \vec{M}_j = \vec{x}_j\cdot P \in \mathbb{G}^n$$

***Randomizer* (optional for further randomization):** The randomizer $(P,\vec{Z})$ is randomized with a random scalar $r' \overset{\$}{\leftarrow} \mathbb{Z}_p$:

$$D_0 = r'\cdot P \qquad \vec{D} = r'\cdot\vec{Z}$$

***Proof of Subset Membership*:** The proof, from Tag_j and $\Sigma_{j,0},\Sigma_{j,1}$:

$$\Sigma'_0 = \Sigma_{j,0} + r\cdot\Sigma_{j,1} \qquad \Sigma'_1 = r'\cdot\Sigma_{j,1} \qquad \text{(optional for further randomization)}$$

which are valid signatures of $(P_{\mathcal{S}}, C_0, \vec{C})$ and $(0, D_0, \vec{D})$, respectively, under VK for the tags $\mathsf{Tag}'_0 = (r+1)\cdot\mathsf{Tag}_j$ and $\mathsf{Tag}'_1 = r'\cdot\mathsf{Tag}_j$, respectively. Note that using r and r', the validity proof $\pi_j = (\mathsf{com}_j, \mathsf{proof}_j)$ of Tag_j can be converted and randomized for Tag'_0 and Tag'_1, into $\pi'_0 = (\mathsf{com}', \mathsf{proof}'_0)$ and $\pi'_1 = (\mathsf{com}', \mathsf{proof}'_1)$, both with the same com' part (with a new random μ'):

$$\begin{aligned}
\tau'_{0,1} &= (r+1)\cdot P = C_0 + P & \tau'_{1,1} &= r'\cdot P = D_0\\
\tau'_{0,2} &= (r+1)\cdot \tau_{j,2} & \tau'_{1,2} &= r'\cdot \tau_{j,2}\\
\tau'_{0,3} &= (r+1)\cdot \tau_{j,3} & \tau'_{1,3} &= r'\cdot \tau_{j,3}\\
\hat{C}' &= \hat{C}_j + \mu'\cdot\hat{V}_{1,1} & \hat{D}' &= \hat{D}_j + \mu'\cdot\hat{V}_{1,2}\\
\Theta'_0 &= (r+1)\cdot(\Theta_j + \mu'\cdot P) & \Theta'_1 &= r'\cdot(\Theta_j + \mu'\cdot P)\\
\Psi'_0 &= (r+1)\cdot(\Psi_j + \mu'\cdot \tau_{j,2}) & \Psi'_1 &= r'\cdot(\Psi_j + \mu'\cdot \tau_{j,2})
\end{aligned}$$

***Individual Verifiability* (optional for further randomization):** A pair of $\mathsf{OT\text{-}LH\text{-}Sign}$ signing keys $\mathsf{sk} = (\mathsf{sk}_i) \overset{\$}{\leftarrow} \mathbb{Z}_p^{n+2}$, and $\mathsf{vk} = \mathsf{sk}\cdot\hat{P} = (\mathsf{vk}_i)_i \in \hat{\mathbb{G}}^{n+2}$, and the signatures

$$\sigma_0 = \mathsf{sk}_1\cdot P + \mathsf{sk}_2\cdot C_0 + \sum_{i=1}^{n}\mathsf{sk}_{i+2}\cdot C_i \qquad \sigma_1 = \mathsf{sk}_2\cdot D_0 + \sum_{i=1}^{n}\mathsf{sk}_{i+2}\cdot D_i$$

which satisfy

$$e(\sigma_0,\hat{P}) = e(P,\mathsf{vk}_1)\cdot e(C_0,\mathsf{vk}_2)\cdot\prod_{i=1}^{n} e(C_i,\mathsf{vk}_{i+2}) \qquad e(\sigma_1,\hat{P}) = e(D_0,\mathsf{vk}_2)\cdot\prod_{i=1}^{n} e(D_i,\mathsf{vk}_{i+2})$$

***Non-Malleability* (optional):** The Diffie-Hellman values $W = r\cdot T$ and $W' = r'\cdot T$, for $T = \mathcal{H}(\mathsf{vk}) \in \mathbb{G}$, together with Groth-Sahai proofs $\pi = (\mathsf{com}, \mathsf{proof})$ and $\pi' = (\mathsf{com}', \mathsf{proof}')$ of Diffie-Hellman tuple for (P, T, C_0, W) and (P, T, D_0, W') respectively:

– $\pi = (\mathsf{com}, \mathsf{proof})$:

- commitment of the witness r, $\mathsf{com} = (\hat{C} = r \cdot \hat{V}_{2,1} + \nu \cdot \hat{V}_{1,1}, \hat{D} = r \cdot \hat{V}_{2,2} + \nu \cdot \hat{V}_{1,2})$, for $\nu \xleftarrow{\$} \mathbb{Z}_p$,
- and the pair of proofs $\mathsf{proof} = (\Theta = \nu \cdot P, \Psi = \nu \cdot T)$, which satisfy

$$e(P, \hat{C}) = e(C_0, \hat{V}_{2,1}) \cdot e(\Theta, \hat{V}_{1,1}) \qquad e(P, \hat{D}) = e(C_0, \hat{V}_{2,2}) \cdot e(\Theta, \hat{V}_{1,2})$$
$$e(T, \hat{C}) = e(W, \hat{V}_{2,1}) \cdot e(\Psi, \hat{V}_{1,1}) \qquad e(T, \hat{D}) = e(W, \hat{V}_{2,2}) \cdot e(\Psi, \hat{V}_{1,2})$$

– and $\pi' = (\mathsf{com}', \mathsf{proof}')$:
- commitment of the witness r', $\mathsf{com}' = (\hat{C}' = r' \cdot \hat{V}_{2,1} + \nu' \cdot \hat{V}_{1,1}, \hat{D}' = r' \cdot \hat{V}_{2,2} + \nu' \cdot \hat{V}_{1,2})$, for $\nu' \xleftarrow{\$} \mathbb{Z}_p$,
- and the pair of proofs $\mathsf{proof}' = (\Theta' = \nu' \cdot P, \Psi' = \nu' \cdot T)$, which satisfy

$$e(P, \hat{C}') = e(D_0, \hat{V}_{2,1}) \cdot e(\Theta', \hat{V}_{1,1}) \qquad e(P, \hat{D}') = e(D_0, \hat{V}_{2,2}) \cdot e(\Theta', \hat{V}_{1,2})$$
$$e(T, \hat{C}') = e(W', \hat{V}_{2,1}) \cdot e(\Psi', \hat{V}_{1,1}) \qquad e(T', \hat{D}') = e(W', \hat{V}_{2,2}) \cdot e(\Psi', \hat{V}_{1,2})$$

The sender then outputs the ciphertext $(C_0, \vec{C}) \in \mathbb{G}^{n+1}$, the randomizer $(D_0, \vec{D}) \in \mathbb{G}^{n+1}$, together with the proof that consists of $(\mathsf{Tag}'_0, \mathsf{Tag}'_1, \Sigma'_0, \Sigma'_1) \in \mathbb{G}^{10} \times \hat{\mathbb{G}}^4$ (as we can again omit $\tau'_{0,1} = C_0 + P$ and $\tau'_{1,1} = D_0$), the signatures $(\mathsf{vk}, \sigma_0, \sigma_1) \in \hat{\mathbb{G}}^{n+2} \times \mathbb{G}^2$, and $(W, W', \pi, \pi') \in \mathbb{G}^6 \times \hat{\mathbb{G}}^4$.

Randomization. Before storing the ciphertext, the receiver (optionally) randomizes it, after having verified all the values $(C_0, \vec{C})$, $(D_0, \vec{D})$, $(\mathsf{Tag}'_0, \mathsf{Tag}'_1, \Sigma'_0, \Sigma'_1)$, $(\mathsf{vk}, \sigma_0, \sigma_1)$, and (W, W', π, π'):

***Verification*:** The receiver verifies
- – vk is fresh (no repetition)
- – $D_0 \neq 0$
- – the tags Tag'_0 and Tag'_1 are valid;
- – the signature Σ'_0 is valid on the message $(P_{\mathcal{S}}, C_0, \vec{C})$ and tag Tag'_0 under VK;
- – the signature Σ'_1 is valid on the message $(0, D_0, \vec{D})$ and tag Tag'_1 under VK;
- – the signature σ'_0 is valid on the message $(P, C_0, \vec{C})$ under vk;
- – the signature σ'_1 is valid on the message $(0, D_0, \vec{D})$ under vk;
- – the Diffie-Hellman proofs π and π' are valid on the tuples (P, T, C_0, W) and (P, T, D_0, W'), respectively.

***Ciphertext*:** The receiver randomizes the ElGamal ciphertext $(C_0, \vec{C})$ using a random scalar $s \xleftarrow{\$} \mathbb{Z}_p$, and the randomizer $(D_0, \vec{D})$

$$C'_0 = C_0 + s \cdot D_0 \qquad \vec{C}' = \vec{C} + s \cdot \vec{D}$$

***Signatures*:** The receiver randomizes and adapts the tags/signatures:

$$\tau''_1 = \tau'_{0,1} + s \cdot \tau'_{1,1} = P + C_0 + s \cdot D_0 \qquad \tau''_2 = \tau'_{0,2} + s \cdot \tau'_{1,2} \qquad \tau''_3 = \tau'_{0,3} + s \cdot \tau'_{1,3}$$
$$\hat{C}'' = \hat{C} + \mu'' \cdot \hat{V}_{1,1} \qquad \hat{D}'' = \hat{D} + \mu'' \cdot \hat{V}_{1,2}$$
$$\Theta'' = \Theta_0 + s \cdot \Theta_1 + \mu'' \cdot \tau''_1 \qquad \Psi'' = \Psi_0 + s \cdot \Psi_1 + \mu'' \cdot \tau''_2$$
$$\Sigma'' = \Sigma'_0 + s \cdot \Sigma'_1 \qquad \sigma = \sigma_0 + s \cdot \sigma_1$$

***Diffie-Hellman Proof*:** The receiver adapts the Groth-Sahai proofs π and π', using s, for $W'' = W + s \cdot W'$, into a randomized π'', with a new $\nu'' \xleftarrow{\$} \mathbb{Z}_p$: $\pi'' = (\mathsf{com}'', \mathsf{proof}'')$:

$$\mathsf{com}'' = (\quad \hat{C}'' = \hat{C} + s \cdot \hat{C}' + \nu'' \cdot \hat{V}_{1,1}, \quad \hat{D}'' = \hat{D} + s \cdot \hat{D}' + \nu'' \cdot \hat{V}_{1,2} \quad)$$
$$\mathsf{proof}'' = (\quad \Theta'' = \Theta + s \cdot \Theta' + \nu'' \cdot P, \quad \Psi'' = \Psi + s \cdot \Psi' + \nu'' \cdot T \quad)$$

The receiver then stores the ciphertext $(C'_0, \vec{C}')$, together with the proof that consists of $(\mathsf{Tag}'', \Sigma'')$, the signature (vk, σ), and (W'', π''), which can be publicly verified.

B.2 Correctness

In this section, we show that all the steps are correct.

Initialisation. The tags $(\tau_{j,1} = P, \tau_{j,2} = t_j \cdot P, \tau_{j,3} = t_j^2 \cdot P)$, with $t_j \overset{\$}{\leftarrow} \mathbb{Z}_p$, for $j = 1, \dots, N$, are made verifiable with $\pi_j = (\mathsf{com}_j, \mathsf{proof}_j)$, for $\mathsf{com}_j = (\hat{C}_j = t_j \cdot \hat{V}_{2,1} + \mu_j \cdot \hat{V}_{1,1}, \hat{D}_j = t_j \cdot \hat{V}_{2,2} + \mu_j \cdot \hat{V}_{1,2})$, for $\mu_j \overset{\$}{\leftarrow} \mathbb{Z}_p$, and $\mathsf{proof} = (\Theta_j = \mu_j \cdot P, \Psi_j = \mu_j \cdot \tau_{j,2})$:

$$\begin{aligned}
e(P, \hat{C}_j) &= e(P, t_j \cdot \hat{V}_{2,1} + \mu_j \cdot \hat{V}_{1,1}) = e(P, t_j \cdot \hat{V}_{2,1}) \cdot e(P, \mu_j \cdot \hat{V}_{1,1}) \\
&= e(t_j \cdot P, \hat{V}_{2,1}) \cdot e(\mu_j \cdot P, \hat{V}_{1,1}) = e(\tau_{j,2}, \hat{V}_{2,1}) \cdot e(\Theta_j, \hat{V}_{1,1}) \\
e(P, \hat{D}_j) &= e(P, t_j \cdot \hat{V}_{2,2} + \mu_j \cdot \hat{V}_{1,2}) = e(P, t_j \cdot \hat{V}_{2,2}) \cdot e(P, \mu_j \cdot \hat{V}_{1,2}) \\
&= e(t_j \cdot P, \hat{V}_{2,2}) \cdot e(\mu_j \cdot P, \hat{V}_{1,2}) = e(\tau_{j,2}, \hat{V}_{2,2}) \cdot e(\Theta_j, \hat{V}_{1,2}) \\
e(\tau_{j,2}, \hat{C}_j) &= e(\tau_{j,2}, t_j \cdot \hat{V}_{2,1} + \mu_j \cdot \hat{V}_{1,1}) = e(\tau_{j,2}, t_j \cdot \hat{V}_{2,1}) \cdot e(\tau_{j,2}, \mu_j \cdot \hat{V}_{1,1}) \\
&= e(t_j \cdot \tau_{j,2}, \hat{V}_{2,1}) \cdot e(\mu_j \cdot \tau_{j,2}, \hat{V}_{1,1}) = e(\tau_{j,3}, \hat{V}_{2,1}) \cdot e(\Psi_j, \hat{V}_{1,1}) \\
e(\tau_{j,2}, \hat{D}_j) &= e(\tau_{j,2}, t_j \cdot \hat{V}_{2,2} + \mu_j \cdot \hat{V}_{1,2}) = e(\tau_{j,2}, t_j \cdot \hat{V}_{2,2}) \cdot e(\tau_{j,2}, \mu_j \cdot \hat{V}_{1,2}) \\
&= e(t_j \cdot \tau_{j,2}, \hat{V}_{2,2}) \cdot e(\mu_j \cdot \tau_{j,2}, \hat{V}_{1,2}) = e(\tau_{j,3}, \hat{V}_{2,2}) \cdot e(\Psi_j, \hat{V}_{1,2})
\end{aligned}$$

That can be verified by four non-zero random scalars $u_1 = \alpha\gamma$, $u_2 = \alpha\delta$, $u_3 = \beta\gamma$, $u_4 = \beta\delta$:

$$\begin{aligned}
e(P, \hat{C}_j)^{u_1} = e(P, \hat{C}_j)^{\alpha\gamma} = e(P^\alpha, \hat{C}_j^\gamma) &= e(\tau_{j,2}, \hat{V}_{2,1})^{\alpha\gamma} \cdot e(\Theta_j, \hat{V}_{1,1})^{\alpha\gamma} = e(\tau_{j,2}^\alpha, \hat{V}_{2,1}^\gamma) \cdot e(\Theta_j^\alpha, \hat{V}_{1,1}^\gamma) \\
e(P, \hat{D}_j)^{u_2} = e(P, \hat{D}_j)^{\alpha\delta} = e(P^\alpha, \hat{D}_j^\delta) &= e(\tau_{j,2}, \hat{V}_{2,2})^{\alpha\delta} \cdot e(\Theta_j, \hat{V}_{1,2})^{\alpha\delta} = e(\tau_{j,2}^\alpha, \hat{V}_{2,2}^\delta) \cdot e(\Theta_j^\alpha, \hat{V}_{1,2}^\delta) \\
e(\tau_{j,2}, \hat{C}_j)^{u_3} = e(\tau_{j,2}, \hat{C}_j)^{\beta\gamma} = e(\tau_{j,2}^\beta, \hat{C}_j^\gamma) &= e(\tau_{j,3}, \hat{V}_{2,1})^{\beta\gamma} \cdot e(\Psi_j, \hat{V}_{1,1})^{\beta\gamma} = e(\tau_{j,3}^\beta, \hat{V}_{2,1}^\gamma) \cdot e(\Psi_j^\beta, \hat{V}_{1,1}^\gamma) \\
e(\tau_{j,2}, \hat{D}_j)^{u_4} = e(\tau_{j,2}, \hat{D}_j)^{\beta\delta} = e(\tau_{j,2}^\beta, \hat{D}_j^\delta) &= e(\tau_{j,3}, \hat{V}_{2,2})^{\beta\delta} \cdot e(\Psi_j, \hat{V}_{1,2})^{\beta\delta} = e(\tau_{j,3}^\beta, \hat{V}_{2,2}^\delta) \cdot e(\Psi_j^\beta, \hat{V}_{1,2}^\delta)
\end{aligned}$$

Hence

$$\begin{aligned}
e(P^\alpha, \hat{C}_j^\gamma \hat{D}_j^\delta) &= e(\tau_{j,2}^\alpha, \hat{V}_{2,1}^\gamma \hat{V}_{2,2}^\delta) \cdot e(\Theta_j^\alpha, \hat{V}_{1,1}^\gamma \hat{V}_{1,2}^\delta) \\
e(\tau_{j,2}^\beta, \hat{C}_j^\gamma \hat{D}_j^\delta) &= e(\tau_{j,3}^\beta, \hat{V}_{2,1}^\gamma \hat{V}_{2,2}^\delta) \cdot e(\Psi_j^\beta, \hat{V}_{1,1}^\gamma \hat{V}_{1,2}^\delta)
\end{aligned}$$

Which means

$$e(P^\alpha \tau_{j,2}^\beta, \hat{C}_j^\gamma \hat{D}_j^\delta) = e(\tau_{j,2}^\alpha \tau_{j,3}^\beta, \hat{V}_{2,1}^\gamma \hat{V}_{2,2}^\delta) \cdot e(\Theta_j^\alpha \Psi_j^\beta, \hat{V}_{1,1}^\gamma \hat{V}_{1,2}^\delta)$$

Ciphertext with Proof of Subset Membership. On $\vec{M} = \vec{M}_j = \vec{x}_j \cdot P \in \mathbb{G}^n$, the ciphertext and the randomizer admit valid signatures, for random $r, r' \overset{\$}{\leftarrow} \mathbb{Z}_p$:

$$\begin{aligned}
C_0 &= r \cdot P + 0 & D_0 &= r' \cdot P \\
\vec{C} &= r \cdot \vec{Z} + \vec{M} & \vec{D} &= r' \cdot \vec{Z} \\
\Sigma_0' &= r \cdot \Sigma_{j,1} + \Sigma_{j,0} & \Sigma_1' &= r' \cdot \Sigma_{j,1}
\end{aligned}$$

under VK for the equivalent tags (for the same t_j), granted linearity

$$\begin{aligned}
\tau_{0,1}' &= (r+1) \cdot P = C_0 + P & \tau_{1,1}' &= r' \cdot P = D_0 \\
\tau_{0,2}' &= (r+1) \cdot \tau_{j,2} = (r+1) \cdot t_j \cdot P = t_j \cdot \tau_{0,1}' & \tau_{1,2}' &= r' \cdot \tau_{j,2} = r' \cdot t_j \cdot P = t_j \cdot \tau_{1,1}' \\
\tau_{0,3}' &= (r+1) \cdot \tau_{j,3} = (r+1) \cdot t_j^2 \cdot P = t_j^2 \cdot \tau_{0,1}' & \tau_{1,3}' &= r' \cdot \tau_{j,3} = r' \cdot t_j^2 \cdot P = t_j^2 \cdot \tau_{1,1}'
\end{aligned}$$

that admit validity proofs, for a random $\mu' \overset{\$}{\leftarrow} \mathbb{Z}_p$

$$\begin{aligned}
\hat{C}' &= \hat{C}_j + \mu' \cdot \hat{V}_{1,1} & \hat{D}' &= \hat{D}_j + \mu' \cdot \hat{V}_{1,2} \\
&= t_j \cdot \hat{V}_{2,1} + (\mu_j + \mu') \cdot \hat{V}_{1,1} & &= t_j \cdot \hat{V}_{2,2} + (\mu_j + \mu') \cdot \hat{V}_{1,2} \\
\Theta_0' &= (r+1) \cdot (\Theta_j + \mu' \cdot P) & \Theta_1' &= r' \cdot (\Theta_j + \mu' \cdot P) \\
&= (r+1) \cdot (\mu_j + \mu') \cdot P & &= r' \cdot (\mu_j + \mu') \cdot P \\
&= (\mu_j + \mu') \cdot \tau_{0,1}' & &= (\mu_j + \mu') \cdot \tau_{1,1}' \\
\Psi_0' &= (r+1) \cdot (\Psi_j + \mu' \cdot \tau_{j,2}) & \Psi_1' &= r' \cdot (\Psi_j + \mu' \cdot \tau_{j,2}) \\
&= (r+1) \cdot (\mu_j + \mu') \cdot \tau_{j,2} & &= r' \cdot (\mu_j + \mu') \cdot \tau_{j,2} \\
&= (\mu_j + \mu') \cdot \tau_{0,2}' & &= (\mu_j + \mu') \cdot \tau_{1,2}'
\end{aligned}$$

Randomization. During randomization, the receiver uses a random scalar $s \xleftarrow{\$} \mathbb{Z}_p$:

$$C'_0 = C_0 + s \cdot D_0 = (r + sr') \cdot P = r'' \cdot P \qquad \vec{C}' = \vec{C} + s \cdot \vec{D} = (r + sr') \cdot \vec{Z} + \vec{M} = r'' \cdot \vec{Z} + \vec{M}$$

and can adapt the signatures:

$$\Sigma'' = \Sigma'_0 + s \cdot \Sigma'_1 \qquad \qquad \sigma = \sigma_0 + s \cdot \sigma_1$$

for the tags:

$$\begin{aligned}
&\tau''_1 = \tau'_{0,1} + s \cdot \tau'_{1,1} = P + C_0 + s \cdot D_0 = P + C'_0 \\
&\tau''_2 = \tau'_{0,2} + s \cdot \tau'_{1,2} = t \cdot (\tau'_{0,1} + s \cdot \tau'_{1,1}) = t \cdot \tau''_1 && \tau''_3 = \tau'_{0,3} + s \cdot \tau'_{1,3} = t \cdot (\tau'_{0,2} + s \cdot \tau'_{1,2}) = t \cdot \tau''_2 \\
&\hat{C}'' = \hat{C} + \mu'' \cdot \hat{V}_{1,1} = t \cdot \hat{V}_{2,1} + (\mu + \mu'') \cdot \hat{V}_{1,1} && \hat{D}'' = \hat{D} + \mu'' \cdot \hat{V}_{1,2} = t \cdot \hat{V}_{2,2} + (\mu + \mu'') \cdot \hat{V}_{1,2} \\
&\Theta'' = \Theta_0 + s \cdot \Theta_1 + \mu'' \cdot \tau''_1 && \Psi'' = \Psi_0 + s \cdot \Psi_1 + \mu'' \cdot \tau''_2 \\
&\quad = \mu(\tau'_{0,1} + s \cdot \tau'_{1,1}) + \mu'' \cdot \tau''_1 = (\mu + \mu'') \cdot \tau''_1 && \quad = \mu(\tau'_{0,2} + s \cdot \tau'_{1,2}) + \mu'' \cdot \tau''_2 = (\mu + \mu'') \cdot \tau''_2
\end{aligned}$$

About the final Diffie-Hellman Proof, first, using s, for $W'' = W + s \cdot W'$, for a new $\nu'' \xleftarrow{\$} \mathbb{Z}_p$:

$$\begin{aligned}
\hat{C}'' &= \hat{C} + s \cdot \hat{C}' + \nu'' \cdot \hat{V}_{1,1} = (r + sr') \cdot \hat{V}_{2,1} + (\nu + s\nu' + \nu'') \cdot \hat{V}_{1,1} = r'' \cdot \hat{V}_{2,1} + \tilde{\nu} \cdot \hat{V}_{1,1} \\
\hat{D}'' &= \hat{D} + s \cdot \hat{D}' + \nu'' \cdot \hat{V}_{1,2} = (r + sr') \cdot \hat{V}_{2,2} + (\nu + s\nu' + \nu'') \cdot \hat{V}_{1,2} = r'' \cdot \hat{V}_{2,2} + \tilde{\nu} \cdot \hat{V}_{1,2} \\
\Theta'' &= \Theta + s \cdot \Theta' + \nu'' \cdot P = (\nu + s\nu' + \nu'') \cdot P = \tilde{\nu} \cdot P \\
\Psi'' &= \Psi + s \cdot \Psi' + \nu'' \cdot T = (\nu + s\nu' + \nu'') \cdot T = \tilde{\nu} \cdot T
\end{aligned}$$

which is thus valid for $T = \mathcal{H}(\mathsf{vk})$ and $W'' = (r + sr') \cdot T = r'' \cdot T$.

B.3 Soundness

Let us now show the soundness.

Initialisation. The tags $(\tau_{j,1} = P, \tau_{j,2} = t_j \cdot P, \tau_{j,3} = t_j^2 \cdot P)$, with $t_j \xleftarrow{\$} \mathbb{Z}_p$, for $j = 1, \ldots, N$, are made verifiable with $\pi_j = (\mathsf{com}_j, \mathsf{proof}_j)$, for $\mathsf{com}_j = (\hat{C}_j = t_j \cdot \hat{V}_{2,1} + \mu_j \cdot \hat{V}_{1,1}, \hat{D}_j = t_j \cdot \hat{V}_{2,2} + \mu_j \cdot \hat{V}_{1,2})$, for $\mu_j \xleftarrow{\$} \mathbb{Z}_p$, and $\mathsf{proof} = (\Theta_j = \mu_j \cdot P, \Psi_j = \mu_j \cdot \tau_{j,2})$, where the verification checks for four independent tuples (u_1, u_2, u_3, u_4)

$$\begin{aligned}
&e(P, \hat{C}_j)^{u_1} \cdot e(P, \hat{D}_j)^{u_2} \cdot e(\tau_{j,2}, \hat{C}_j)^{u_3} \cdot e(\tau_{j,2}, \hat{D}_j)^{u_4} \\
&\quad = (e(\tau_{j,2}, \hat{V}_{2,1}) \cdot e(\Theta_j, \hat{V}_{1,1}))^{u_1} \cdot (e(\tau_{j,2}, \hat{V}_{2,2}) \cdot e(\Theta_j, \hat{V}_{1,2}))^{u_2} \\
&\quad \cdot (e(\tau_{j,3}, \hat{V}_{2,1}) \cdot e(\Psi_j, \hat{V}_{1,1}))^{u_3} \cdot (e(\tau_{j,3}, \hat{V}_{2,2}) \cdot e(\Psi_j, \hat{V}_{1,2}))^{u_4}
\end{aligned}$$

linear algebra leads to the four individual above equalities.

Knowing that $\hat{V}_{1,2} = u \cdot \hat{V}_{1,1}$ and $\hat{V}_{2,2} = v \cdot \hat{V}_{2,1}$ for $u \neq v$,

$$\begin{aligned}
e(P, \hat{C}_j)^{-u} \cdot e(P, \hat{D}_j) &= e(\tau_{j,2}, \hat{V}_{2,1})^{-u} \cdot e(\Theta_j, \hat{V}_{1,1})^{-u} \cdot e(\tau_{j,2}, \hat{V}_{2,2}) \cdot e(\Theta_j, \hat{V}_{1,2}) \\
e(\tau_{j,2}, \hat{C}_j)^{-u} \cdot e(\tau_{j,2}, \hat{D}_j) &= e(\tau_{j,3}, \hat{V}_{2,1})^{-u} \cdot e(\Psi_j, \hat{V}_{1,1})^{-u} \cdot e(\tau_{j,3}, \hat{V}_{2,2}) \cdot e(\Psi_j, \hat{V}_{1,2})
\end{aligned}$$

so

$$\begin{aligned}
e(P, \hat{D}_j - u \cdot \hat{C}_j) &= e(\tau_{j,2}, \hat{V}_{2,2} - u \cdot \hat{V}_{2,1}) \cdot e(\Theta_j, \hat{V}_{1,2} - u \cdot \hat{V}_{1,1}) \\
e(\tau_{j,2}, \hat{D}_j - u \cdot \hat{C}_j) &= e(\tau_{j,3}, \hat{V}_{2,2} - u \cdot \hat{V}_{2,1}) \cdot e(\Psi_j, \hat{V}_{1,2} - u \cdot \hat{V}_{1,1})
\end{aligned}$$

hence, for $\hat{V} = \hat{V}_{2,2} - u \cdot \hat{V}_{2,1} \neq 0$ and $\hat{R}_j = \hat{D}_j - u \cdot \hat{C}_j$,

$$e(P, \hat{R}_j) = e(\tau_{j,2}, \hat{V}) \qquad \qquad e(\tau_{j,2}, \hat{R}_j) = e(\tau_{j,3}, \hat{V})$$

which shows that the discrete logarithm of $\tau_{j,2}$ in basis P is the same as the discrete logarithm of $\tau_{j,3}$ in basis $\tau_{j,2}$: $(P, \tau_{j,2}, \tau_{j,3})$ is an SDH tuple.

Ciphertext with Proof of Subset Membership. Before randomization, the receiver verifies

- vk is fresh (no repetition), so $T = \mathcal{H}(\mathsf{vk})$ is truly random in $\mathbb{G}$;
- $D_0 \neq 0$, which will be important for the receipt-freeness;
- the tags Tag'_0 and Tag'_1 are valid, with a common $\mathsf{com}' = (\hat{C}', \hat{D}')$: they are equivalent, with $\tau'_{0,1} = C_0 + P$ and $\tau'_{1,1} = D_0$;
- the signature Σ'_0 is valid on the message $(P_{\mathcal{S}}, C_0, \vec{C})$ and tag Tag'_0 under VK: granted the unforgeability of the signature, $(P_{\mathcal{S}}, C_0, \vec{C})$ necessarily comes from the combination of messages signed under equivalent tags under VK, which are only the pairs in the initial public parameters, $(P_{\mathcal{S}}, 0, \vec{M}_j)$ and $(0, P, \vec{Z})$, for each j, and possibly multiple $P_{\mathcal{S}}$ (if multiple subsets $\mathcal{S}$). This is thus $\alpha \cdot (P_{\mathcal{S}'}, 0, \vec{M}_j) + \beta \cdot (0, P, \vec{Z})$. Necessarily, $P_{\mathcal{S}} = \alpha \cdot P_{\mathcal{S}'}$, where $P_{\mathcal{S}} = \mathcal{H}(\mathcal{S})$ and $P_{\mathcal{S}'} = \mathcal{H}(\mathcal{S}')$ with a random oracle $\mathcal{H}$, the extractability property of the signature breaks the discrete logarithm problem, unless $P_{\mathcal{S}} = P_{\mathcal{S}'}$ and $\alpha = 1$. Then $(C_0, \vec{C})$ is a ciphertext of $\vec{M}_j = \vec{x}_j \cdot P$, for $\vec{x}_j \in \mathcal{S}$. We can thus use $(C_0, \vec{C}) = (r \cdot P, r \cdot \vec{Z} + \vec{M})$;
- the signature Σ'_1 is valid on the message $(0, D_0, \vec{D})$ and tag Tag'_1 under VK, the same analysis holds: because of the 0 in the first components, $(D_0, \vec{D})$ is necessarily a randomization of $(P, \vec{Z})$. We can thus use $(D_0, \vec{D}) = (r' \cdot P, r' \cdot \vec{Z})$. Furthermore, as $D_0 \neq 0$, $r' \neq 0$;
- the signature σ'_0 is valid on the message $(P, C_0, \vec{C})$ under vk, which holds by construction;
- the signature σ'_1 is valid on the message $(0, D_0, \vec{D})$ under vk, which holds by construction;
- the Diffie-Hellman proofs π and π' are valid on the tuples (P, T, C_0, W) and (P, T, D_0, W'), respectively, which means that W and W' are correct Diffie-Hellman values: in the AGM, one needs to know r and r' to compute $W = r \cdot T$ and $W' = r' \cdot T$, for a fresh T. Only the current voter can do. One cannot replay another vote, with unknown randomness.

Randomization. After randomization, the receiver outputs $(C'_0, \vec{C}')$ and $(\mathsf{Tag}'', \Sigma'')$ and (vk, σ). The latter σ being valid on $(P, C'_0, \vec{C}')$ under vk means that $(P, C'_0, \vec{C}')$ can only be on a linear combination of $(P, C_0, \vec{C})$ and $(0, P, \vec{Z})$ signed by the voter. Because of P in the first component, under the unforgeability of the signature scheme, $(C'_0, \vec{C}')$ is a randomization of $(C_0, \vec{C})$.

The former signature Σ'' being valid on $(P_{\mathcal{S}}, C'_0, \vec{C}')$ under VK and a valid tag convinces everybody the plaintext is in $\mathcal{S}$.

The validity of π'' means that W'' is correct Diffie-Hellman value: in the AGM, one needs to know r'' to compute $W'' = r'' \cdot T$, for a fresh T, which is only possible for the voter (with r and r') and the receiver (with s) together. Another voter cannot have helped the receiver to generate such a valid W''.