Post-Quantum Privacy for Traceable Receipt-Free Encryption

Paola de Perthuis[1] and Thomas Peters[2]

[1] Centrum Wiskunde & Informatica (CWI), Nederland
[2] Université Catholique de Louvain (UCLouvain), Belgique

Abstract. Traceable Receipt-free Encryption (TREnc) has recently been introduced as a verifiable public-key encryption primitive endowed with a unique security model. In a nutshell, TREnc allows randomizing ciphertexts in transit in order to remove any subliminal information up to a public trace that ensures the non-malleability of the underlying plaintext. A remarkable property of TREnc is the indistinguishability of the *randomization of chosen ciphertexts* against traceable chosen-ciphertext attacks (TCCA). The main application lies in voting systems by allowing voters to encrypt their votes, tracing whether a published ballot takes their choices into account, and preventing them from proving how they voted. While being a very promising primitive, the few existing TREnc mechanisms solely rely on discrete-logarithm related assumptions making them vulnerable to the well-known record-now/decrypt-later attack in the wait of quantum computers.

We address this limitation by building the first TREnc whose privacy withstands the advent of quantum adversaries in the future. To design our construction, we first generalize the original TREnc primitive that is too restrictive to be easily compatible with built-in lattice-based semantically-secure encryption. Our more flexible model keeps all the ingredients generically implying receipt-free voting. Our instantiation relies on Ring Learning With Errors (RLWE) with pairing-based statistical zero-knowledge simulation sound proofs from Groth-Sahai, and further enjoys a public-coin common reference string removing the need of a trusted setup.

1 Introduction

After its first definition in 2022 [DPP22], Traceable Receipt-free ENCryption (TREnc) has moved from the initial theoretical construction to very recently a more practical one [DPP24] removing the need for a trusted setup and offering tighter reductions, along with a Rust implementation.

TREnc schemes come with a new privacy notion, Indinstinguishability under Traceable Chosen Ciphertext Attacks (TCCA), which extends the more classical CPA and CCA notions to one in which, with access to a decryption oracle, and ciphertexts marked with a tag, the adversary chooses two ciphertexts with a same tag in the challenge phase (instead of cleartexts in the usual notions), and receives a randomization of one of them, for which she must guess the original ciphertext. After receiving the challenge, she still has access to a decryption oracle, but now only for ciphertexts with a tag different from the challenge one.

This unique feature makes it easier to design voting system with receipt-freeness, where voters are unable to convince anyone how they voted as defined in [CCFG16,DPP22]. Recently in [DPP24], the TREnc approach has been shown to provide the shortest ballot size in verifiable receipt-free elections.

A Post-Quantum Security Tradeoff. Although the cryptographic techniques allowing to build receipt-free voting systems became more and more practical, the most efficient solutions solely rely on hard discrete-log (DLog) groups. This situation is uncomfortable since an adversary who records all the communication between the talliers and the data available on the bulletin board during an election today can keep this information until the advent of quantum computer to break the privacy later. Even if [DPP22] provides a generic construction of TREnc, it is not known how to instantiate all the building blocks without relying on pairing-based tools whose security requires the hardness of computing discrete logarithms.

As a first step toward designing an efficient post-quantum (PQ) receipt-free election without trusted parameter generation, the current work provides the first TREnc with public-coin parameters which whistands the above "harvest-now, decrypt-later" attack in the sense that the resulting election scheme will remain secure as long as it ends before the availability of a quantum computer. Indeed, as building trustworthy post-quantum receipt-free eVoting remains very challenging, we observe that it is enough to hide the message/vote using PQ encryption scheme where it has to be computationally hidden only (since decryption must still hold) and to rely on pre-quantum tools to provide all the additional properties of the TREnc as long as they only carry message-related information that are statistically hidden.

This observation even allows us to select well-understood tools whose security relies on common standard assumptions. More precisely, we can encrypt the message/vote with a lattice-based scheme with a module compatible with the size of pairing-based groups, where we can adapt the existing SXDH-based tools of previous TREnc constructions with the algebraic structure of the RLWE ciphertext. As usual, SXDH means that DDH, the decisional Diffie-Hellman assumption, holds in both source groups, and RLWE refers to the Ring version of the decisional Learning With Error assumption. The reason why this approach solves the "harvest-now, decrypt-later" issue is because it is sufficient that the TREnc properties hold until the end of the election, after which no additional decryption (query) will happen. From that point, the only remaining attack path for the adversary will be to break the security of the PQ encryption.

We stress that this solution is not (fully) post-quantum since the pre-quantum tools include zero-knowledge proofs whose soundness holds under DLog related assumption which can be exploit by a quantum adversary to fake a verifiable ciphertext. Therefore, in the wait of new techniques to design efficient PQ receipt-free voting systems, it is safer to replace pre-quantum TREnc by our new construction. Such a priorization of security properties has already been done in previous works (for example, in [BdPP23] for a Key-Encapsulation Mechanism).

Open Problems for a Totally Post-Quantum Solution. In the current knowledge, building a TREnc requires at least two ingredients additionally to a randomizable semantically-secure encryption with a linear structure: a (tag-based) randomizable (simulation-sound) zero-knowledge (ZK) proof to ensure that ciphertexts have been honestly computed, and a linearly-only homomorphic signature scheme used to control the malleability of the ciphertexts, i.e., to prevent anything beyond rerandomization, except in the security proof. More precisely, the latter component leads to the traceability notion of the TREnc ciphertext, which means that no efficient adversary (even knowing the decryption key) can modify the underlying plaintext and keep the same trace of an honestly computed ciphertext while mainting its public verifiability. This security notion tells that even the authorities of an election cannot modify the content of ballots computed by honest voters from a TREnc.

The post-quantum literature provides solutions to linearly-only homomorphic signature for binary fields [BF11], or solutions with both additive and multiplicative homomorphism [GVW15]. Assuming restricting the homomorphism be feasible, none of these constructions allow adapting one signature among two and at the same time hiding which input signature has been modified. That is, while these schemes could be adapted to comply with the traceability requirement of a TREnc, they would reveal which ciphertext has been randomized, which prevents the TCCA notion to hold. Instead of trying to construct ad hoc signatures with both properties, one may rely on additional ZK proof on the top of them to confer this missing hiding property. However, it is unclear today how one can build the appropriate efficient post-quantum proof system.

For the TCCA property, we need a (simulation-sound) ZK proof that also enyoys a malleability property to follow the randomization of ciphertexts and the hiding property explained above ensuring that we cannot detect the orinal proof that has been modified (for a same tag). The lattice-based ZK proof [LNP22] does not have all these properties nor [BS22]. The situation is even worse if one would like the TREnc ciphertexts to ensure that plaintexts satisfy complex relations as it is often the case in a voting application.

Simpler Simulation-Sound Proofs for TREnc. Our construction makes use of a simpler simulation-sound proof technique than the one of [DPP24], which was based on the Ràfols branching technique on Groth-Sahai (GS) proofs [Ràf15]. Simulation-Sound proofs have the advantage of making the generation of proofs of false statements hard for an attacker, even when she has received simulated proofs for false statements of her choice, which will be important in the ciphertext privacy security game, in which an attacker, even after receiving a ciphertext with simulated proofs in the challenge phase, should not be able to generated new simulated ciphertexts with the same trace, and ask for their decryption.

Simulation soundness through OR proofs was already exibited in [Gro06, CKLM12], but with different security notions, in which these proofs were not randomizable and were used in constructions with privacy notions in which plaintexts are used as a challenge, rather than ciphertexts as in TCCA. Moreover, in our construction, all proofs are associated to a tag; however, in the TCCA security proof, the adversary may send as a challenge two ciphertexts associated with a tag for which she has already queried a decryption (and this property will be useful to attain receipt freeness in the subsequent voting scheme). This means that the public key may not be programmed to embed the tag of the challenge ciphertexts.

In [DPP24], the OR proofs are done by generating an additive share of a Groth-Sahai Common Reference String (CRS), in which if the original CRS is perfectly binding, then at least one of the shared CRSs is, resulting in proofs for which the randomization, also redistributing these shares, are not conceptually simple. Our simulation-sound proofs rely on a Linearly-Homomorphic Structure Preserving signature scheme, already using in the rest of the protocol for tracing properties, but now used in the OR proof, relying on the fact that signatures of null vectors are trivial to generate, and signing either the original statement or a tag-specific vector, in the case of simulated proofs.

Generalizing TREnc Security Notions. Our construction also generalizes the initial definitions of TREnc to a context in which the distribution of ciphertexts changes after the entity in charge of granting the Traceable-Chosen Ciphertext Attacks (TCCA) security by a redistribution of their randomness has handled them; indeed, previous constructions from ElGamal ciphertexts did not raise this question as the randomized ciphertexts followed the same noise distribution as fresh ones, but updating the randomness of RLWE-based ciphertexts without knowing their decryption key generally leads to an augmentation of their noise levels; a process which is irreversible without using the bootstrapping techniques requiring circular security that are implemented in Fully-Homomorphic Encryption schemes. In this line, we also stress the differences between the TCCA notion and a Randomizability one in which the added noise would statistically hide the old one (using noise flooding techniques, which is also achievable with the construction described here). To achieve this, perfectly hiding GS proofs are performed on all the bits used in the ciphertext construction and randomization, also allowing a fine-grained control on the message space, and naturally the inclusion of the proof of any statement on the encrypted message, which is a new functionality for TREnc constructions.

Our generalization of TREnc notions also requires a new definition of their ciphertext privacy, as with distinct randomized and fresh ciphertexts spaces, it is not directly implied by the TCCA-security anymore.

A Public-Coin CRS Generation. Finally, using GS proofs on group elements committed using common-reference string elements drawn using public randomness, in the perfectly witness-indistinguishable mode of this proof system, allows us to remove the need for a trusted setup, which may not be realistic in real-life scenarios. As the current construction uses a public-coin perfectly witness-indistinguishable Groth-Sahai common reference string, it generalizes the TREnc notion of verifiability in a way in which, though the normal key derivation does not allow the verification of ciphertexts being in the range of correct encryptions and randomizations, there exists an indistinguishable key generation algorithm which comes with a trapdoor allowing this verification; previous works exploited particular cases in which the range could easily be verified, but this is not generally true, as captured by this more universal definition.

Technical Overview

Simulation-Sound Proofs from LHSP Signatures. One building-block of this work's construction is a new, statistically hiding, simulation-sound proof from Linearly-Homomorphic Structure-Preserving (LHSP) signatures and Groth-Sahai commitments, which is simpler than [DPP24]'s based on [Ràf15], which required a Groth-Sahai CRS randomization.

This new simulation-sound proof works in the following way: when setting up the system, two group elements are selected using public-coin randomness, and will be used as an LHSP scheme's public key, as well as other group elements that will be used as a Groth-Sahai common reference string (CRS), that will perfectly hide committed group elements.

From there, to each proof is associated a tag τ, and a vector $\boldsymbol{v}_\tau \leftarrow (G, \tau G)$, where G is a public group generator, and an integer $\mathfrak{b}$ multiplying the generator G is committed using a Groth-Sahai perfectly witness-indistinguishable CRS. Then, the proof statement will be of the following form, denoting $\boldsymbol{Q}(\boldsymbol{X}, \boldsymbol{\mathfrak{X}}) = (q_1(\boldsymbol{X}, \boldsymbol{\mathfrak{X}}), \ldots, q_n(\boldsymbol{X}, \boldsymbol{\mathfrak{X}})) = (0, \ldots, 0)$ the original statement on group elements forming a solution in the variables $(\boldsymbol{X}, \boldsymbol{\mathfrak{X}})$, that is made simulation-sound with our framework: "I know a signature on $(1 - \mathfrak{b})(G, \tau G)$ and $\mathfrak{b} \cdot \boldsymbol{Q}(\boldsymbol{X}, \boldsymbol{\mathfrak{X}}) = (0, \ldots, 0)$.".

It is trivial to sign a null vector with an LHSP signature scheme, whatever the public key, so honest proofs will be made setting $\mathfrak{b} \leftarrow 1$, and indeed prove the knowledge of a solution to the system $\boldsymbol{Q}(\boldsymbol{X}, \boldsymbol{\mathfrak{X}}) = (0, \ldots, 0)$. However, simulated proofs will use a perfectly indistinguishable CRS generation, in which the simulator will keep an LHSP secret key with respect to the public key in the CRS. With this key, she will be able to use $\mathfrak{b} \leftarrow 0$ in the proof, signing $\boldsymbol{v}_\tau$ while proving a trivial statement $(0, \ldots, 0) = (0, \ldots, 0)$ in the other part of the proof. As LHSP signatures can only be reused to sign linear combinations of known signed vectors, the adversary will not be able to reuse this simulated proof for new tags, under the Symmetric eXternal Diffie-Hellman (SXDH) assumption, making it simulation-sound.

Building a TREnc Scheme on Top of Lattice-Based Ciphertexts. Randomizing Learning With Errors (LWE) (or Ring-LWE) based ciphertexts generally introduces a change in the distribution of their randomness with respect to fresh ones; in fact, the randomness introduced should be carefully controlled, as when it overlaps a threshold, it may change the result of the decryption. In this work, the noise is decomposed into bits, used to perform a linear combination of public elements to generate the ciphertexts; this linear encryption operation, and the fact that committed witnesses are bits, are proven in the Groth-Sahai framework, using vectors of group elements whose exponents are (R)LWE ciphertext components, to be perfectly witness-indistinguishable and randomizable. This also leads to a natural functionality allowing to additionally provide the system with the proof of any kind of statement on the encrypted messages, which is a new TREnc functionality that is relevant for voting applications.

Moreover, in previous TREnc constructions, the encryptor would sign several vectors to allow the simulated proof to generate a signature for any ciphertext; however, in our contruction, because of the noise growth in (R)LWE ciphertexts, the additional rows would allow teh randomizer to change the ciphertext decryption. We thus resort to zero-knowledge proofs of a valid signature rather than sending signatures in the clear, which allows the simulation to be performed for the challenge ciphertext in the privacy security games. These signatures then become directly simulatable.

2 Preliminaries

2.1 Notations

Vectors will be denoted with bold letters, such as $\boldsymbol{v}$, and be vertical unless stated otherwise. For $\boldsymbol{v} = (v_1, \ldots, v_n)$, $\boldsymbol{v}[i]$ will denote v_i. $\mathbf{0}_n$ will denote the vector of zeros of length n. $\boldsymbol{e}_{n,i}$ will denote the i-th vector of the canonical basis of a vectorial space of dimension n: in short, the vector with zeros in all coordinates but the i-th, which is equal to one. $\|$ will be used to denote the concatenation of two vectors: for instance $\boldsymbol{u}\|\boldsymbol{v}$ will be the concatenation of vectors $\boldsymbol{u}$ and $\boldsymbol{v}$. $\boldsymbol{u} \odot \boldsymbol{v} = (u_1v_1, \ldots, u_nv_n)$ will denote the pointwise product of vectors $\boldsymbol{u} = (u_1, \ldots, u_n)$ and $\boldsymbol{v} = (v_1, \ldots, v_n)$, and $\langle \boldsymbol{u}; \boldsymbol{v} \rangle$ their inner-product. Matrices will generally be underlined, such as with $\underline{M}$. The identity matrix of dimension n will be denoted as $\underline{\mathsf{Id}}_n$.

Group elements will be denoted with capital letters. For G an element of an additive group $\mathbb{G}$ of order q, $\boldsymbol{u}G$ will denote the vector $(u_1G, \ldots, u_nG) \in \mathbb{G}^n$.

A *pairing setting* $(p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e, G, \mathfrak{G})$ will describe two additive groups $\mathbb{G}$ and $\hat{\mathbb{G}}$ of order p, with G and $\mathfrak{G}$ two respective generators, and a bilinear pairing operation $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$ going into the multiplicative group of order p $\mathbb{G}_T$ generated by $e(G, \mathfrak{G})$.

For any quotient ring $\mathcal{R} = \mathbb{Z}_q[X]/\mathsf{r}(X)$, with $\mathsf{r} \in \mathbb{Z}_q[X]$ of degree n, the function $\mathsf{pol}_{\mathcal{R}} : \mathbb{Z}_q^n \to \mathcal{R}$ will associate, to any $\boldsymbol{v} = (v_0, \ldots, v_{n-1}) \in \mathbb{Z}_q^n$, the corresponding polynomial $\sum_{i=0}^{n-1} v_i X^i \in \mathcal{R}$.

Moreover, for any x in $\mathbb{Z}_q$, $[x]_q$ will denote the (or a if there are two) smallest representative of x in $\mathbb{Z}$ in absolute value.

For any integers $a \leqslant b$, $[\![a; b]\!]$ will denote the set: $\{x \in \mathbb{Z} | a \leqslant x \leqslant b\}$. Given a finite set $\mathcal{S}$, $x \xleftarrow{\$} \mathcal{S}$ will mean that x is sampled from the uniform distribution $\mathcal{U}_{\mathcal{S}}$ on $\mathcal{S}$. Given two distributions $\mathcal{D}_0$ and $\mathcal{D}_1$, and a Probabilistic Polynomial Time

(PPT) adversary $\mathcal{A}$, her distinguishing advantage on these distributions will be defined as: $\mathsf{Adv}_{\mathcal{A}}^{\mathcal{D}_0,\mathcal{D}_1} = \left|\Pr_{x \xleftarrow{\$} \mathcal{D}_0}\{\mathcal{A}(x) = 0\} - \Pr_{x \xleftarrow{\$} \mathcal{D}_1}\{\mathcal{A}(x) = 0\}\right|$.

IND-CPA security will be attained, for a public-key encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\mathcal{M}$, when for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, $\mathcal{A}$'s probability of winning the security game defined in figure 1 (*i. e.*, having it output 1), is negligibly close to one half in the security parameter λ.

```
$\mathsf{Exp}_{\mathcal{A}}^{\mathsf{CPA}}(\lambda)$:

$(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$
$(m_0, m_1, \mathsf{st}) \xleftarrow{\$} \mathcal{A}_1(\mathsf{pk})$
$b \xleftarrow{\$} \{0; 1\}$
**if** $m_0 \notin \mathcal{M}$ or $m_1 \notin \mathcal{M}$ **then** return 0
$c^* \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}, m_b)$, $b' \xleftarrow{\$} \mathcal{A}_2(c^*, \mathsf{st})$
**if** $b' = b$ return 1, **else** return 0
```

Fig. 1. IND-CPA security experiment

2.2 Hard Problems

Our construction will rely on the hardness of classical cryptographic problems; the Chosen Plaintext Attack (CPA) privacy of encrypted messages will rely on the Learning With Errors (LWE) one, stated hereafter:

Definition 1 (The Learning With Errors (LWE) Average-Case Decision Assumption). *states, with respect to $q, n \in \mathbb{N}$, and an error distribution χ, that the two following distributions are computationally hard to distinguish:*

$$\mathcal{D}_0 = \left\{(\boldsymbol{a}, \langle \boldsymbol{a}; \boldsymbol{s}\rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q | \boldsymbol{a}, \boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n, e \xleftarrow{\$} \chi\right\}$$
$$\mathcal{D}_1 = \left\{(\boldsymbol{a}, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_q | \boldsymbol{a} \xleftarrow{\$} \mathbb{Z}_q^n, b \xleftarrow{\$} \mathbb{Z}_q\right\};$$

this statement is expressed with respect to any Probabilistic Polynomial Time (PPT) adversary $\mathcal{A}$ and security parameter $\lambda \in \mathbb{N}$, as: $\mathsf{Adv}_{\mathcal{A}}^{D-\mathsf{LWE}}(\lambda) = \mathsf{negl}(\lambda)$, where $\mathsf{Adv}_{\mathcal{A}}^{D-\mathsf{LWE}}(\lambda)$ denotes $\mathcal{A}$'s advantage, when receiving an element of $\mathcal{D}_\beta$ for $\beta \xleftarrow{\$} \{0; 1\}$, in guessing the value of β.

and more precisely, on its cyclotomic ring variant:

Definition 2 (The Ring-LWE Average-Case Decision Assumption). *states, with respect to q, $n \in \mathbb{N}$, $\mathsf{r} \in \mathbb{Z}_q[X]$ a cyclotomic polynomial of degree n, and an error distribution χ on $\mathcal{R}$, where $\mathcal{R} \leftarrow \mathbb{Z}_q[X]/\mathsf{r}(X)$, that the two following distributions are computationally hard to distinguish:*

$$\mathcal{D}_0 = \left\{(\mathsf{a}, \mathsf{a} \cdot \mathsf{s} + \mathsf{e}) \in \mathcal{R}^2 | \mathsf{a}, \mathsf{s} \xleftarrow{\$} \mathcal{R}, \mathsf{e} \xleftarrow{\$} \chi\right\} \quad \mathcal{D}_1 = \left\{(\mathsf{a}, \mathsf{b}) \in \mathcal{R}^2 | \mathsf{a}, \mathsf{b} \xleftarrow{\$} \mathcal{R}\right\};$$

this statement is expressed with respect to any Probabilistic Polynomial Time (PPT) adversary $\mathcal{A}$ and security parameter $\lambda \in \mathbb{N}$, as: $\mathsf{Adv}_{\mathcal{A}}^{D-\mathsf{RLWE}}(\lambda) = \mathsf{negl}(\lambda)$, where $\mathsf{Adv}_{\mathcal{A}}^{D-\mathsf{RLWE}}(\lambda)$ denotes $\mathcal{A}$'s advantage, when receiving an element of $\mathcal{D}_b$ for $b \xleftarrow{\$} \{0; 1\}$, in guessing the value of b.

The Traceability and Traceable Chosen Ciphertext Attack (TCCA) security of the scheme will rely on the SXDH assumption, presented hereafter:

Definition 3 (The Decisional Diffie-Hellman (DDH) Assumption). *states, with respect to a group $(\mathbb{G}, +)$ of prime order p, that given one of its generators, G, the two following distributions are computationally hard to distinguish:*

$$\mathcal{D}_0 = \left\{(aG, bG, abG) | a, b \xleftarrow{\$} \mathbb{Z}_p\right\} \qquad \mathcal{D}_1 = \left\{(aG, bG, cG) | a, b, c \xleftarrow{\$} \mathbb{Z}_p\right\};$$

this statement is expressed with respect to any PPT adversary $\mathcal{A}$ and security parameter $\lambda \in \mathbb{N}$, as: $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{DDH},\mathbb{G}}(\lambda) = \mathsf{negl}(\lambda)$, where $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{DDH},\mathbb{G}}(\lambda)$ denotes $\mathcal{A}$'s advantage, when receiving an element of $\mathcal{D}_b$ for $b \xleftarrow{\$} \{0; 1\}$, in guessing the value of b.

Definition 4 (The Symmetric eXternal Diffie-Hellman (SXDH) Assumption). *states, with respect to two additive groups of primer order p, $\mathbb{G}$ and $\hat{\mathbb{G}}$, and a bilinear pairing operation $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$ mapping elements into the multiplicative group $\mathbb{G}_T$ of order p, the DDH assumption is true both in $\mathbb{G}$ and in $\hat{\mathbb{G}}$; this statement is expressed with respect to any PPT adversary $\mathcal{A}$ and security parameter $\lambda \in \mathbb{N}$, as: $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{SXDH}}(\lambda) = \mathsf{negl}(\lambda)$, where $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{SXDH}}(\lambda) = \max\{\mathsf{Adv}_{\mathcal{A}}^{\mathsf{DDH},\mathbb{G}}(\lambda), \mathsf{Adv}_{\mathcal{A}}^{\mathsf{DDH},\hat{\mathbb{G}}}(\lambda)\}$.*

Finally, the TCCA security will also rely on the resistance of hash functions against collisions, a property stated here:

Definition 5 (Collision Resistance). *A family of functions $\mathcal{F}_h = \{h_k : \{0; 1\}^{n(k)} \to \{0; 1\}^{m(k)}\}_k$ lists collision-resistant hash functions if for any k, $n(k) \geqslant m(k)$, and there exists a PPT algorithm* Sampl *outputting, on input a security parameter $\lambda \in \mathbb{N}$, h_k in the family, such that for any PPT adversary $\mathcal{A}$:*

$$\Pr\left\{ \{x \neq y\} \cap \{h_k(x) = h_k(y)\} \left| \begin{array}{c} h_k \xleftarrow{\$} \mathsf{Sampl}(1^\lambda) \\ (x, y) \xleftarrow{\$} \mathcal{A}(h_k, 1^\lambda) \end{array} \right. \right\} \leqslant \mathsf{negl}(\lambda).$$

2.3 An RLWE-Based Ciphertext Instantiation

A simple example of such a scheme would be, stating with the FV scheme [FV12] with multiplicative depth zero, defining the plaintext space $\mathcal{R}_t = \mathbb{Z}_t[X]/(X^n+1)$, with n a power of two, and the ciphertext space $\mathcal{R}_p^2$ with $\mathcal{R}_p = \mathbb{Z}_p[X]/(X^n+1)$, $\sigma \in]0; 1[$ a noise parameter, $\Delta \leftarrow \lfloor q/t \rfloor$, and $\Gamma = (q, t, n, \sigma)$ the parameter set. χ_s will denote the gaussian distribution on $\mathcal{R}_p$ with standard deviation s.

$\mathsf{KeyGen}(1^\lambda, \Gamma) \to (\mathsf{sk}, \mathsf{pk})$**:** samples $\mathsf{a} \xleftarrow{\$} \mathcal{R}_p$, and $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n$, sets $\mathsf{s} \leftarrow \mathsf{pol}_{\mathcal{R}_p}(\boldsymbol{s})$, samples $\mathsf{e} \xleftarrow{\$} \chi_\sigma$, sets: $(\mathsf{p}, \mathsf{p}') \leftarrow ([-(\mathsf{a} \cdot \mathsf{s} + \mathsf{e})]_q, \mathsf{a}) \in \mathcal{R}_p^2$, $\mathsf{pk} \leftarrow (\mathsf{p}, \mathsf{p}', \Delta, \sigma)$ and $\mathsf{sk} \leftarrow \mathsf{s}$, and returns $(\mathsf{sk}, \mathsf{pk})$.

$\mathsf{Enc}_{\mathsf{pk}}(\mathsf{m}) \to (\mathsf{c}, \mathsf{c}')$**:** samples $\mathsf{e}_1, \mathsf{e}_2 \xleftarrow{\$} \chi_\sigma$, $\boldsymbol{u} \in \{0; 1\}^n$, sets $\mathsf{u} \leftarrow \mathsf{pol}_{\mathcal{R}_p}(\boldsymbol{u})$, and returns: $(\mathsf{c}, \mathsf{c}') \leftarrow ([\mathsf{p} \cdot \mathsf{u} + \mathsf{e}_1 + \Delta \cdot [\mathsf{m}]_t]_q, [\mathsf{p}' \cdot \mathsf{u} + \mathsf{e}_2]_q)$. The gaussian distribution χ_σ may also be given as an optional argument to the encryption function.

$\mathsf{Dec}_{\mathsf{sk}}((\mathsf{c}, \mathsf{c}')) \to \mathsf{m}$**:** computes: $\mathsf{d} \leftarrow [\mathsf{c} + \mathsf{s} \cdot \mathsf{c}']$, and returns: $\mathsf{m} \leftarrow [\lfloor \mathsf{d}/\Delta \rceil]_t$.

The ciphertexts in this scheme can be added with linear homorphism, up to a certain bound. For any x in $\mathbb{Z}_a$, $[x]_a$ will denote its representative in $\llbracket -\lceil \frac{a}{2} \rceil ; \lfloor \frac{a}{2} \rfloor \rrbracket$, when applied to a vector it will denote the operation applied to each of its components, and for any x in $\mathcal{R}_a$, $[\mathsf{x}]_a$ will denote the representative of x reduced by the quotient polynomial ($X^n + 1$ in our case) of $\mathcal{R}_a$ with coefficients in $\llbracket -\lceil \frac{a}{2} \rceil ; \lfloor \frac{a}{2} \rfloor \rrbracket$.

In [FV12], the authors show that this scheme grants semantic security from RLWE, even if $\boldsymbol{s}$ is drawn from $\{0;1\}^n$ rather than over $\mathbb{Z}_q^n$ optimizing the size of the secret key [ACPS09, LPR10]. The statistical correctness in also shown in [FV12].

2.4 Linearly-Homomorphic Structure-Preserving Signatures

These signatures consist of two group elements signing a vector with components in the same group (first primitives stemming from [AFG$^+$10, AHO10]), with the additional property (from [LPJY14]) that a linear combination of signatures will yield a signature on the corresponding linear combination of vectors. The algorithms of such an LHSP scheme are recalled hereafter:

$\mathsf{KeyGen}(\mathsf{pp}, n) \to (\mathsf{pk}, \mathsf{sk})$**:** on input the public parameters pp describing additive groups of primer order p, $\mathbb{G}$ and $\hat{\mathbb{G}}$, generators $\mathfrak{G}$ and $\mathfrak{H}$ of $\hat{\mathbb{G}}$, and a bilinear pairing operation $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$ mapping elements into the multiplicative group $\mathbb{G}_T$ of order p, and the vector length $n \in \mathbb{N}$ (of polynomial size), this algorithm draws $\chi_1, \ldots \chi_n, \gamma_1, \ldots, \gamma_n \xleftarrow{\$} \mathbb{Z}_p$, sets, for each index i in $\llbracket 1; n \rrbracket$, $\mathfrak{G}_i \leftarrow \chi_i \mathfrak{G} + \gamma_i \mathfrak{H}$, and then the secret key to: $\mathsf{sk} \leftarrow (((\chi_i, \gamma_i))_{i \in \llbracket 1;n \rrbracket}, \mathsf{pp})$, and the public key to $\mathsf{pk} \leftarrow ((\mathfrak{G}_1, \ldots, \mathfrak{G}_n), \mathsf{pp})$, finally outputting: $(\mathsf{pk}, \mathsf{sk})$.

$\mathsf{Sign}(\mathsf{sk}, (M_1, \ldots, M_n)) \to \sigma$**:** on input sk parsed as an output of KeyGen and $(M_1, \ldots, M_n) \in \mathbb{G}^n$, the algorithm sets and returns: $\sigma \leftarrow (\Sigma_1, \Sigma_2) \leftarrow (\sum_{i=1}^{n} \chi_i M_i, \sum_{i=1}^{n} \gamma_i M_i)$.

$\mathsf{SignDer}(\mathsf{pk}, (\omega_i)_{i=1}^{m}, (\sigma_i)_{i=1}^{m}) \to \sigma$**:** on input pk parsed as an output of KeyGen, $(\omega_1, \ldots, \omega_m) \in \mathbb{Z}_p^m$, for a natural m, and the σ_i's parsed as outputs of Sign, this algorithm derives a signature on the linear combination with weights ω_i of vectors they sign by outputting: $\sigma \leftarrow (\Sigma_1, \Sigma_2) \leftarrow \sum_{i=1}^{m} \omega_i \sigma_i$.

$\mathsf{Ver}(\mathsf{pk}, \sigma, (M_1, \ldots, M_n)) \to b$**:** parsing pk as a corresponding output of KeyGen, $\sigma = (\Sigma_1, \Sigma_2)$ as an output of Sign, the algorithm outputs $b \leftarrow 1$ if and only if: $e(\Sigma_1, \mathfrak{G}) e(\Sigma_2, \mathfrak{H}) = \prod_{i=1}^{n} e(M_i, \mathfrak{G}_i)$; else, it outputs $b \leftarrow 0$.

2.5 The Groth Sahai Proof System

Provided by Groth and Sahai's seminal work in [GS08], this proof system, in a commit and prove framework, allows the randomization of commitments and proofs. Furthermore, it grants witness-indistinguishable proofs of quadratic relations (on scalars, groups elements, or a mix of both, in a pairing setting), and can be used in two indistinguishable modes, one of which leads to perfectly binding and the other to perfectly witness-indistinguishable proofs.

In this work, Groth-Sahai (GS) algorithms will be used with a public-coin generation in the perfectly witness-indistinguishable case for the commitment of group elements. They are presented in the case of proofs on group elements in Appendix A.

3 Generalizing TREnc

In our construction, the use of lattice-based ciphertexts implies that their randomized version will not have the same noise distribution as their fresh counterparts. In previous TREnc constructions, they did, and the TCCA security thus implied privacy of fresh ciphertexts with a weak CCA notion (of CCA security for ciphertexts with an adversarially-chosen tag). This is not the case for the current constructions, and as a consequence, we had to generalize previous TREnc security notions.

Moreover, we underline that the TREnc randomization notion does not need to redistribute ciphertexts' randomness in the whole randomness space, and not even in an exponentially bigger space as the one of fresh ciphertexts, as in usual noise flooding approaches that seek to mask all the noise information to a lattice ciphertext decryptor. In a TREnc scheme, randomization may be done with a simple addition of a fresh encryption of zero, under (R)LWE.

Additionally, in Traceability and Verifiability security notions, the adversary is provided with the secret-key, as these security notions should hold even against authorities in a voting system, in order to ensure the correctness of the results with respect to participants' intentions.

Definition 6 (Traceable Receipt-Free Encryption, extension of [DPP22]). *A* Traceable Receipt-Free Encryption *scheme (TREnc) is a public key encryption scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *augmented with a triple of algorithms* $(\mathsf{Trace}, \mathsf{Rand}, \mathsf{Ver})$*:*

- $\mathsf{Gen}(1^\lambda)$ *generates and outputs a public-secret key pair* $(\mathsf{pk}, \mathsf{sk})$*;*
- $\mathsf{Enc}(\mathsf{pk}, m)$ *is split into two probabilistic sub-algorithms. First, it runs the link key generation algorithm* $\mathsf{LGen}(\mathsf{pk})$ *which outputs an ephemeral secret link key* lk*. Second, it runs the linked encryption algorithm* $\mathsf{LEnc}(\mathsf{pk}, \mathsf{lk}, m)$ *which outputs a ciphertext c encrypting m, and including a trace as defined next;*
- $\mathsf{Trace}(\mathsf{pk}, c)$ *is a public algorithm that returns a trace t on input a ciphertext c.*
- $\mathsf{Rand}(\mathsf{pk}, c)$ *partially randomizes the ciphertext c and returns a ciphertext c';*
- $\mathsf{Ver}(\mathsf{pk}, c, \ell)$ *outputs 1 if the ciphertext c is deemed valid according to the context* $\ell \in \{\mathsf{fresh}, \mathsf{rand}\}$*, and 0 otherwise.*

The message space $\mathcal{M}$ is implicitly defined by the public key pk*. By definition, the ciphertext space* $\mathcal{C}_{\mathsf{fresh}}$ *is the image of $\mathcal{M}$ by* $\mathsf{Enc}(\mathsf{pk}, \cdot)$*, and similarly, the ciphertext space* $\mathcal{C}_{\mathsf{rand}}$ *is the image of* $\mathcal{C}_{\mathsf{fresh}}$ *by* $\mathsf{Rand}(\mathsf{pk}, \cdot)$*. The public key* pk *can be made implicit everywhere when it is identifiable from the context.*

A TREnc must satisfy several correctness conditions: (*Link traceability*) For every pk in the range of Gen, every lk in the range of $\mathsf{LGen}(\mathsf{pk})$, the encryptions of every pair of messages (m_0, m_1) trace to "each other", that is, it always holds that $\mathsf{Trace}(\mathsf{pk}, \mathsf{LEnc}(\mathsf{pk}, \mathsf{lk}, m_0)) = \mathsf{Trace}(\mathsf{pk}, \mathsf{LEnc}(\mathsf{pk}, \mathsf{lk}, m_1))$; (*Publicly Traceable Randomization*) For every pk in the range of Gen, every message m and every c in the range of $\mathsf{Enc}(\mathsf{pk}, m)$, we have that $\mathsf{Dec}(\mathsf{sk}, c) = \mathsf{Dec}(\mathsf{sk}, \mathsf{Rand}(\mathsf{pk}, c))$ and $\mathsf{Trace}(\mathsf{pk}, c) = \mathsf{Trace}(\mathsf{pk}, \mathsf{Rand}(\mathsf{pk}, c))$; (*Honest verifiability*) For every pk in the range of Gen, every messages m, and every ciphertext $c \in \mathcal{C}_{\mathsf{fresh}}$, it holds that $\mathsf{Ver}(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, m), \mathsf{fresh}) = 1$ and $\mathsf{Ver}(\mathsf{pk}, \mathsf{Rand}(\mathsf{pk}, c), \mathsf{rand}) = 1$.

In the original definition $\mathcal{C}_{\mathsf{fresh}}$ and $\mathcal{C}_{\mathsf{rand}}$ are equal, and Ver is independent of the context $\ell \in \{\mathsf{fresh}, \mathsf{rand}\}$. The novel general case given here also allows these spaces to be disjoint and even to easily recognize that a valid ciphertext for one context does not belong to ciphertext space of the other context, as in our new construction. The main generalization of the primitive comes to define Rand for a *one-time* execution while, in the original syntax, Rand can still be applied serially on its outputs. In [DPP22], a TREnc further comes with a strong randomization property which, while elegant, is not needed to keep the essence of the notions allowing to generically build a receipt-free voting system.

We now turn to the security model satisfied by a (general) TREnc. We start with the verifiability that we extend to the different contexts $\ell \in \{\mathsf{fresh}, \mathsf{rand}\}$. Intuitively, it should be hard given sk to produce a valid ciphertext c for one context such that c is not in the corresponding ciphertext space $\mathcal{C}_\ell$. That is, there must exist some message m, some link key lk, and some coins that can explain c as a run of the appropriate algorithms even if they are not easily computable. However, to prove the verifiability criteria, one often needs an efficient way to check if the adversary is successful or not. Unlike [DPP22], we thus explicitly require the existence of this algorithm in the definition.

Definition 7 (Verifiability, modified from [DPP22]). *A TREnc is* verifiable *if it exists efficient* SimGen *and* Check *such that:*

1. $\{(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)\} \approx_c \{(\mathsf{pk}, \mathsf{sk}) \mid (\mathsf{pk}, \mathsf{sk}, \mathsf{tk}) \leftarrow \mathsf{SimGen}(1^\lambda)\}$*;*
2. *For any* $(\mathsf{pk}, \mathsf{sk}, \mathsf{tk}) \leftarrow \mathsf{SimGen}(1^\lambda)$ *and any context* $\ell \in \{\mathsf{fresh}, \mathsf{rand}\}$*, we have* $\mathsf{Check}(\mathsf{tk}, \cdot, \ell) \in \{0, 1\}$*, and for all* c*,* $\Pr[\mathsf{Check}(\mathsf{tk}, c, \ell)) = 0 \wedge c \in \mathcal{C}_\ell] = \mathsf{negl}(\lambda)$*;*
3. *For every PPT adversary* $\mathcal{A}$*,* $\Pr[\,\mathsf{Ver}(\mathsf{pk}, c, \ell) = 1 \wedge \mathsf{Check}(\mathsf{tk}, c, \ell) = 0 \mid (\mathsf{pk}, \mathsf{sk}, \mathsf{tk}) \leftarrow \mathsf{SimGen}(1^\lambda), c \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{sk})] = \mathsf{negl}(\lambda)$*.*

The traceability notion ensures the orignal encryptor of a message that any (randomized) ciphertext with the same trace contain the same message even against the decryptor (as long as the encryptor uses the link key generated by LGen a single time). This notion is particularly usefull in a voting system where voters keep track of their randomized ballots while being sure that the authorities cannot alter their votes. We slightly generalized the notion due to [DPP22] by granted the adversary with an oracle that produces fresh ciphertexts on input a plaintext. This allows for a more general learning phase, and the adversary is successful if it can produce a ciphertext that traces to one of the returned oracle's ciphertexts while decrypting to another message than the original. The earlier notion reduces to a single query.

Definition 8 (Traceability, extended from [DPP22]). *A TREnc is* traceable *if for every PPT adversary $\mathcal{A}$, the experiment $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Trace}}(\lambda)$ defined in figure 3 returns 1 with a negligible probability in λ. The traceability advantage is defined as $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Trace}}(\lambda) = \Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Trace}}(\lambda) = 1]$.*

We stress that if an adversary is able to produce a ciphertext c^* such that $\mathsf{Ver}(\mathsf{pk}, c^*, \mathsf{fresh}) = 1$ in place of $\mathsf{Ver}(\mathsf{pk}, c^*, \mathsf{rand}) = 1$ in the traceable experiment, it can simply output $c^* \leftarrow \mathsf{Rand}(\mathsf{pk}, c^*)$ to win the game.

```
$\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Trace}}(\lambda)$:
--------------------------------------------------
$(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$
$\mathcal{L} \leftarrow \varnothing$
$c^* \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_{\mathsf{Enc}}}(\mathsf{sk})$
**if** $\exists (m, c) \in \mathcal{L} : \mathsf{Trace}(\mathsf{pk}, c) = \mathsf{Trace}(\mathsf{pk}, c^*)$
    and $\mathsf{Ver}(\mathsf{pk}, c^*, \mathsf{rand}) = 1$
    and $\mathsf{Dec}(\mathsf{sk}, c^*) \neq m$
    **then** return 1
**else** return 0.
```

Fig. 2. The traceability experiment. On input a message m_i, the oracle $\mathcal{O}_{\mathsf{Enc}}$ returns $c_i \leftarrow \mathsf{Enc}(\mathsf{pk}, m_i)$ and updates $\mathcal{L} \leftarrow \mathcal{L} \cup \{(m_i, c_i)\}$.

The privacy notion of the original TREnc is the indistinguishability of the *randomization* of adversarially-chosen valid *ciphertexts* that trace to each other with access to a decryption oracle. This is the TCCA notion which deviates from most of the existing game-based privacy notions whose indistinguishability is defined by *encrypting* adversarially chosen *messages*. The TCCA notion implies that any subliminal information maliciously added to an adversarially computed ciphertext does not help the adversary to distinguish which ciphertext has been processed by Rand. By encrypting a vote with a TCCA TREnc, the voter is unable to explain the content of its randomized version, hence the receipt-freeness.

Definition 9 (TCCA, adapted from [DPP22]). *A TREnc is TCCA secure if for every PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the experiment $\mathsf{Exp}_{\mathcal{A}}^{TCCA}(\lambda)$ defined in Figure 3 returns 1 with a probability negligibly close to $\frac{1}{2}$ in λ, meaning that $\mathcal{A}$'s advantage in distinguishing b is negligible in λ.*

In the TCCA and wCCA security games, the adversary is allowed to use a challenge tag which is equal to a tag of a ciphertext that was previously queried to the decryption oracle (though such a tag may not be queried to the oracle after receiving the challenge); this detail is important when transforming the TREnc construction into a voting scheme, in order to attain receipt-freeness, the notion preventing participants from sellingn their votes. Indeed, in the security games, the TCCA adversary will need to simulate an election result taking challenge ciphertexts into account without the decryption key, and will achieve this by querying the decryption of ciphertexts with the same tag beforehand.

This notion says nothing about the privacy of the encryption of chosen messages. Up to now, Enc could be the identity function or any function leaking its input. However, in a voting system, for instance, the ballot privacy should hold against the randomizing server that is only trusted for the receipt-freeness. In [DPP22], it has been shown that a TREnc that is also strongly randomizable when $\mathcal{C}_{\mathsf{fresh}} = \mathcal{C}_{\mathsf{rand}}$ automatically provides the privacy of the encryption. That

```
$\mathsf{Exp}_{\mathcal{A}}^{\mathrm{TCCA}}(\lambda)$:

$(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$
$(c_0, c_1, \mathsf{st}) \xleftarrow{\$} \mathcal{A}_1^{\mathsf{Dec}(\cdot)}(\mathsf{pk})$
$b \xleftarrow{\$} \{0; 1\}$
if $\mathsf{Trace}(\mathsf{pk}, c_0) \neq \mathsf{Trace}(\mathsf{pk}, c_1)$
    or $\mathsf{Ver}(\mathsf{pk}, c_0, \mathsf{fresh}) \neq 1$
    or $\mathsf{Ver}(\mathsf{pk}, c_1, \mathsf{fresh}) \neq 1$
    then return 0
$c^* \xleftarrow{\$} \mathsf{Rand}(\mathsf{pk}, c_b)$
$b' \xleftarrow{\$} \mathcal{A}_2^{\mathsf{Dec}^*(\cdot)}(c^*, \mathsf{st})$
if $b' = b$ return 1, else return 0
```

```
$\mathsf{Exp}_{\mathcal{A}}^{\mathsf{wCCA}}(\lambda)$:

$(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$
$(m_0, m_1, \mathsf{lk}, \mathsf{st}) \xleftarrow{\$} \mathcal{A}_1^{\mathsf{Dec}(\cdot)}(\mathsf{pk})$
$b \xleftarrow{\$} \{0; 1\}$
if $\mathsf{lk} \notin \mathsf{LGen}(1^\lambda)$
    or $m_0 \notin \mathcal{M}$ or $m_1 \notin \mathcal{M}$
    then return 0
$c^* \xleftarrow{\$} \mathsf{LEnc}(\mathsf{pk}, \mathsf{lk}, m_b)$
$b' \xleftarrow{\$} \mathcal{A}_2^{\mathsf{Dec}^*(\cdot)}(c^*, \mathsf{st})$
if $b' = b$ return 1, else return 0
```

Fig. 3. TCCA and wCCA security experiments; $\mathcal{A}_2$ has access to a decryption oracle $\mathsf{Dec}^*(\cdot)$ which returns a decryption of any input ciphertext c such that $\mathsf{Trace}(\mathsf{pk}, c) \neq \mathsf{Trace}(\mathsf{pk}, c^*)$, and that there exists $\ell \in \{\mathsf{fresh}, \mathsf{rand}\}$ such that $\mathsf{Ver}(\mathsf{pk}, c, \ell) = 1$, and returns $\perp$ for any input ciphertext not meeting this condition, as well as to a $\mathsf{Dec}(\cdot)$ oracle doing exactly the same, but without the condition on the queries' trace.

is because the randomization fully redistribute the ciphertext among those that have the same encryted message and the same trace. Therefore, after encrypting a message we can always indistinguishably randomize it and rely on the TCCA security. Since in general a TREnc does not necessarily satisfy this property, it must come with an additional privacy notion for the fresh ciphertexts. Here, we adopt the adaptive-tag weak CCA notion of [MRY04] defined for tag-based encryption, and naturally adapt it to our syntax as adaptive-trace weak CCA security of TREnc.

Definition 10 (wCCA, adapted from [MRY04]). *A TREnc is adaptive-trace weakly CCA secure if for every PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the experiment $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{wCCA}}(\lambda)$ defined in Figure 3 returns 1 with a probability negligibly close to $\frac{1}{2}$ in λ, meaning that $\mathcal{A}$'s advantage in distinguishing b is negligible in λ.*

While a selective-trace notion of wCCA might be enough in some application, the TCCA notion already requires an adaptive-trace flavor as the trace is chosen by the adversary when it sends c_0 and c_1 at the beginning of the challenge phase.

4 An LWE-based TREnc Scheme

This description uses the Groth-Sahai scheme, denoted GS, the LHSP signature scheme, as well as an FV post-quantum Public-Key Encryption scheme PQPKE (though it may naturally be generalized to any LWE-based scheme).

4.1 Initialization Algorithm Gen

Input: the security parameter $\lambda \in \mathbb{N}$.

Computations: picks, with public coin randomness, a pairing setting $\mathsf{pp} = (p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e, G, \mathfrak{G}) \xleftarrow{\$} \mathsf{GS.Setup}(1^\lambda)$, for which the SXDH assumption is assumed to hold with at least λ bits of security. The algorithm then draws $H \xleftarrow{\$} \mathbb{G}$, $\mathfrak{H} \xleftarrow{\$} \hat{\mathbb{G}}$, $\boldsymbol{\varphi} = (\boldsymbol{U}_1, \boldsymbol{U}_2, \boldsymbol{\mathfrak{U}}_1, \boldsymbol{\mathfrak{U}}_2) \xleftarrow{\$} \mathsf{GS.HCRSGen}(\mathsf{pp})$, $\mathcal{H}$ a collision-free function mapping elements to $\mathbb{Z}_p$, $\mathsf{spk} \leftarrow (\mathfrak{G}_{\mathsf{spk},1}, \mathfrak{G}_{\mathsf{spk},2}) \xleftarrow{\$} \hat{\mathbb{G}}^2$, and sets: $\mathsf{crs} \leftarrow (\mathsf{pp}, \boldsymbol{\varphi}, H, \mathfrak{H}, \mathsf{spk}, \mathcal{H})$.

Then, it generates, with $t \leftarrow 2$, $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{PQPKE.KeyGen}(1^\lambda, (p, t, n, \sigma))$, where the FV parameters (n, σ) are taken with respect to p defined in pp to give an efficient scheme with a security on λ bits for noise terms drawn from χ_σ while allowing the decryption of ciphertext with a noise drawn from $2\chi_\sigma$ to be correct except with negligible probability in λ: denoting B the smallest natural such that except with negligible probability, elements drawn from χ_σ are in $[\![B]\!]$, ciphertexts with noises e, e' in $\mathsf{pol}_{\mathcal{R}_p}([\![2B]\!])$ will always be decrypted correctly.

Output: the algorithm sets $\mathsf{PK} \leftarrow (\mathsf{pk}, \mathsf{crs})$ and returns: $(\mathsf{PK}, \mathsf{sk})$.

4.2 Encryption Algorithm Enc

Input: PK, parsed as an output of Gen, and $\boldsymbol{m} = (m_1, \ldots, m_n) \in \{0; 1\}^n$.

Post-Quantum Ciphertext Generation An encryption of $\mathsf{m} = \mathsf{pol}_{\mathcal{R}_t}(\boldsymbol{m}) \in \mathsf{pol}_{\mathcal{R}_t}(\{0; 1\}^n) \subset \mathcal{R}_t$ is computed drawing $\boldsymbol{u} = (u_1, \ldots, u_n) \xleftarrow{\$} \{0; 1\}^n, \mathsf{e}, \mathsf{e}' \xleftarrow{\$} \chi_\sigma$ and, parsing pk as $(\mathsf{p}, \mathsf{p}', \Delta, \sigma)$, as:

$$(\mathsf{c}, \mathsf{c}') \leftarrow (\mathsf{p} \cdot \mathsf{u} + \Delta \cdot [\mathsf{m}]_t + \mathsf{e}, \mathsf{p}' \cdot \mathsf{u} + \mathsf{e}');$$

denoting $\mathsf{a} \mapsto \mathsf{pol}^{-1}_{\mathcal{R}_p}(\mathsf{a})$ the function which, given $\mathsf{a} \in \mathcal{R}_p$, returns $\boldsymbol{a} \in \mathbb{Z}_p^n$ such that: $\mathsf{a}(X) = \langle \boldsymbol{a}; (1, X, \ldots, X^{n-1}) \rangle$, and naming: $\boldsymbol{p} = (p_1, \ldots, p_n) \leftarrow \mathsf{pol}^{-1}_{\mathcal{R}_p}(\mathsf{p}), \boldsymbol{p}' = (p'_1, \ldots, p'_n) \leftarrow \mathsf{pol}^{-1}_{\mathcal{R}_p}(\mathsf{p}'), \boldsymbol{e} = (e_1, \ldots, e_n) \leftarrow \mathsf{pol}^{-1}_{\mathcal{R}_p}(\mathsf{e}), \boldsymbol{e}' = (e'_1, \ldots, e'_n) \leftarrow \mathsf{pol}^{-1}_{\mathcal{R}_p}(\mathsf{e}), \boldsymbol{c} = (c_1, \ldots, c_n) \leftarrow \mathsf{pol}^{-1}_{\mathcal{R}_p}(\mathsf{c}), \boldsymbol{c}' = (c'_1, \ldots, c'_n) \leftarrow \mathsf{pol}^{-1}_{\mathcal{R}_p}(\mathsf{c}')$, one has, as $\mathcal{R}_p$ is a cyclotomic ring:

$$\boldsymbol{c} = \left(\sum_{k=1}^{n} p_k u_{[1-k]_n+1} + \Delta m_1 + e_1, \ldots, \sum_{k=1}^{n} p_k u_{n+1-k} + \Delta m_n + e_n \right)$$
$$\boldsymbol{c}' = \left(\sum_{k=1}^{n} p'_k u_{[1-k]_n+1} + e'_1, \ldots, \sum_{k=1}^{n} p'_k u_{n+1-k} + e'_n \right)$$

and $\boldsymbol{c} \| \boldsymbol{c}'$ may, except with negligible probability in λ (if this unlucky event happens, then the algorithm aborts), be expressed as the bit decomposition of $(m_1, \ldots, m_n, u_1, \ldots, u_n, e_1, \ldots, e_n, e'_1, \ldots, e'_n)^T$, denoted $\boldsymbol{w} = (w_1, \ldots, w_N)^T$, with $N = 2n\lfloor \log_2(B) \rfloor$, times a public matrix $\underline{P}$ of $\mathbb{Z}_p^{2n(1+\lfloor \log_2(B) \rfloor)}$ deterministically determined by pk:

$$\underline{P} \leftarrow \left(\begin{array}{cccccc}
\Delta & 0 & & \cdots & & 0 \\
\vdots & \ddots & \ddots & \ddots & & \vdots \\
0 & \cdots & \Delta & 0 & \cdots & 0 \\
p_1 & \cdots & p_n & p'_1 & \cdots & p'_n \\
\vdots & \ddots & & & \ddots & \vdots \\
p_2 & \cdots & p_1 & p'_2 & \cdots & p'_1 \\
2^{\lfloor \log_2(B) \rfloor} & 0 & \cdots & & & 0 \\
2^{\lfloor \log_2(B) \rfloor - 1} & 0 & \cdots & & & \\
\vdots & \vdots & & & & \vdots \\
1 & 0 & \cdots & & & \\
0 & 2^{\lfloor \log_2(B) \rfloor} & 0 & \cdots & & \\
\vdots & \vdots & \vdots & & & \vdots \\
 & 1 & 0 & \cdots & & 0 \\
\vdots & & \ddots & \ddots & & \vdots \\
 & & \cdots & \cdots & 0 & 2^{\lfloor \log_2(B) \rfloor} \\
\vdots & \vdots & \vdots & & & \vdots \\
0 & & \cdots & \cdots & 0 & 1
\end{array} \right).$$

The algorithm sets: $B_{\mathsf{fresh}} \leftarrow \lfloor \log_2(B) \rfloor$, $N \leftarrow 2n(1 + B_{\mathsf{fresh}})$, and names $\underline{P}$'s rows: $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_N \in \mathbb{Z}_p^{2n}$.

Tracing and Validity Simulation-Sound Proofs Denoting, for $B_{\mathsf{Rand}} = B_{\mathsf{fresh}}$, $\widetilde{N} \leftarrow n(1 + 2B_{\mathsf{Rand}})$, the following matrix of $\mathbb{Z}_p^{\widetilde{N} \times 2n}$:

$$\underline{\widetilde{P}} \leftarrow \left(\begin{array}{cccccc}
p_1 & \cdots & p_n & p'_1 & \cdots & p'_n \\
\vdots & \ddots & & & \ddots & \vdots \\
p_2 & \cdots & p_1 & p'_2 & \cdots & p'_1 \\
2^{B_{\mathsf{Rand}}} & 0 & \cdots & & & 0 \\
2^{B_{\mathsf{Rand}} - 1} & 0 & \cdots & & & \\
\vdots & \vdots & & & & \vdots \\
1 & 0 & \cdots & & & \\
0 & 2^{B_{\mathsf{Rand}}} & 0 & \cdots & & \\
\vdots & \vdots & \vdots & & & \vdots \\
 & 1 & 0 & \cdots & & 0 \\
\vdots & & \ddots & \ddots & & \vdots \\
 & & \cdots & \cdots & 0 & 2^{B_{\mathsf{Rand}}} \\
\vdots & \vdots & \vdots & & & \vdots \\
0 & & \cdots & \cdots & 0 & 1
\end{array} \right),$$

determined deterministically from pk, naming $\underline{\widetilde{P}}$'s rows: $\widetilde{\boldsymbol{p}}_1, \ldots, \widetilde{\boldsymbol{p}}_{\widetilde{N}} \in \mathbb{Z}_p^{2n}$, the algorithm proceeds with the following steps:

1. the algorithm sets: $\mathfrak{b} \leftarrow 1$ (meaning that the proof will not be simulated);
2. it then generates the one-time signature key pair: $(\mathsf{osk}, \mathsf{opk}) \xleftarrow{\$} \mathsf{LHSP.KeyGen}((\mathsf{pp}, \mathfrak{H}), 2n + 1 + \widetilde{N})$;
3. denoting:

$$\underline{T} \leftarrow \begin{pmatrix} \boldsymbol{c}^T & \boldsymbol{c}'^T & 1 & 0 & \dots & & 0 \\ \multicolumn{2}{c}{\widetilde{\boldsymbol{p}}_1^T} & 0 & 1 & \ddots & & \vdots \\ \multicolumn{2}{c}{\vdots} & \vdots & & \ddots & \ddots \quad \ddots & \vdots \\ \multicolumn{2}{c}{\vdots} & \vdots & & & \ddots \quad \ddots & 0 \\ \multicolumn{2}{c}{\widetilde{\boldsymbol{p}}_{\widetilde{N}}^T} & 0 & & \dots & 0 & 1 \end{pmatrix} \cdot G$$

for each i in $\left[\!\left[1; 1 + \widetilde{N}\right]\!\right]$, the algorithm signs the i-th row $\boldsymbol{T}_i$ of $\underline{T}$, with:

$$\sigma_i \leftarrow \mathsf{LHSP.Sign}(\mathsf{osk}, \mathfrak{b}\boldsymbol{T}_i);$$

4. then, in the Groth-Sahai framework, and setting the tag $\tau \leftarrow \mathcal{H}(\mathsf{opk}, \mathsf{PK})$, it generates a proof of knowledge of a solution – colored in orange for witnesses in $\mathbb{G}$, and in cyan for witnesses in $\hat{\mathbb{G}}$, and denoting O_1 and O_2 the element O provided in two distinct commitments, to the following system of equations, denoting $\boldsymbol{p}_i = (p_{i,1}, \dots, p_{i,2n})$, $\widetilde{\boldsymbol{p}}_i = (\widetilde{p}_{i,1}, \dots, \widetilde{p}_{i,2n})$, encoded into $\mathcal{E}_{\mathsf{SiSo}}$, as well as $\widetilde{\mathcal{P}} \subset \left[\!\left[1; \widetilde{N}\right]\!\right] \times [\![1; 2n]\!]$ a subset made of indices (i, j) such that every $\widetilde{p}_{i,j} \neq 0$ is present exactly once, and ϕ a function associating to every (i, j) in $\left[\!\left[1; \widetilde{N}\right]\!\right] \times [\![1; 2n]\!]$ such that $\widetilde{p}_{i,j} \neq 0$ the $(\widetilde{i}, \widetilde{j}) \in \widetilde{\mathcal{P}}$ such that $\widetilde{p}_{i,j} = \widetilde{p}_{\widetilde{i},\widetilde{j}}$:

$$\begin{cases} e(c_1 G, \mathfrak{b}\mathfrak{G}) = e(\mathfrak{b}c_1 G, \mathfrak{G}) \\ \vdots \\ e(c'_n G, \mathfrak{b}\mathfrak{G}) = e(\mathfrak{b}c'_n G, \mathfrak{G}) \\ \forall (i,j) \in \widetilde{\mathcal{P}} : e(\widetilde{p}_{i,j} G, \mathfrak{b}\mathfrak{G}) = e(\mathfrak{b}\widetilde{p}_{i,j} G, \mathfrak{G}) \\ e(G, \mathfrak{b}\mathfrak{G}) = e(\mathfrak{b}G, \mathfrak{G}) \\ \\ \forall i \in \left[\!\left[1; 1 + \widetilde{N}\right]\!\right] : \\ e(\sigma_{i,1} G, \mathfrak{G}) e(\sigma_{i,2} G, \mathfrak{H}) = e(\mathfrak{b}G, \mathfrak{G}_{\mathsf{opk},2n+i}) \prod_{\substack{j \in [\![1;2n]\!] \\ \phi(i,j) \neq \perp}} e(\mathfrak{b}t_{\phi(i,j)} G, \mathfrak{G}_{\mathsf{opk},j}) \\ \\ \forall j \in [\![1; n]\!] : \prod_{i=1}^N e(w_i G, \mathfrak{b}\mathfrak{G})^{p_{i,j}} = e(c_j G, \mathfrak{b}\mathfrak{G}) \\ \qquad\qquad\quad \prod_{i=1}^N e(w_i G, \mathfrak{b}\mathfrak{G})^{p_{i,n+j}} = e(c'_j G, \mathfrak{b}\mathfrak{G}) \\ \\ \forall i \in [\![1; N]\!] : e(w_i G, w_i \mathfrak{G}) = e(G, w_i \mathfrak{G}) \\ \qquad\qquad\quad e(w_i G, \mathfrak{G}) = e(G, w_i \mathfrak{G}) \\ \\ e(O_1, \mathfrak{G}) e(O_2, \mathfrak{H}) = e((1 - \mathfrak{b})G, \mathfrak{G}_{\mathsf{spk},1}) e((1 - \mathfrak{b})\tau G, \mathfrak{G}_{\mathsf{spk},2}) \\ e(G, \mathfrak{b}\mathfrak{G}) e((1 - \mathfrak{b})G, \mathfrak{G}) = e(G, \mathfrak{G}) \\ e(\tau G, \mathfrak{b}\mathfrak{G}) e((1 - \mathfrak{b})\tau G, \mathfrak{G}) = e(\tau G, \mathfrak{G}) \end{cases}$$

with:

$$\mathit{\Pi}_{\mathsf{SiSo}} = (\mathsf{Com}, \pi, \mathcal{E}_{\mathsf{SiSo}}) \overset{\$}{\leftarrow} \mathsf{GS.Com\&Pr}(\mathsf{pp}, \boldsymbol{\varphi}, w, \mathcal{E}_{\mathsf{SiSo}}),$$

and denoting: $\mathsf{Com} = \mathsf{Com}_{\mathfrak{B}} \| (\mathsf{Com}_{\sigma,i})_i \| ((\mathsf{Com}_{\boldsymbol{T}_i,j})_j)_i \| \mathsf{Com}_{\mathfrak{B},G} \| \mathsf{Com}_{\boldsymbol{W}} \| \mathsf{Com}_{\mathfrak{W}}$ $\| \mathsf{Com}_{\sigma,\tau} \| \mathsf{Com}_{\tau}$ and $\pi = (\pi_{\mathfrak{B},i,j})_{i \in [\![1;\widetilde{N}]\!], j \in [\![1;2n]\!]} \| (\pi_1, \pi_{\mathfrak{B},k+1})_{k \in [\![1;N]\!]} \| (\pi_{\mathsf{Sign},i})_i$ $\| (\pi_{\mathsf{IP},j})_j \| (\pi_{\mathsf{Bit},k})_{k \in [\![1;N]\!]} \| \pi_{\tau}$.

Output: the algorithm sets the whole TREnc ciphertext to: $C \leftarrow ((\mathsf{c}, \mathsf{c}'), \mathsf{opk}, \mathit{\Pi}_{\mathsf{SiSo}})$, and returns: (osk, C).

4.3 Tracing Algorithm Trace

On input a ciphertext C, parsed as an output of Enc, this algorithm returns opk.

4.4 Randomization Algorithm Rand

Input: PK parsed as an output of Gen, and C parsed as an output of Enc.

Randomized Ciphertext Generation To randomize $(\mathsf{c}, \mathsf{c}')$, the algorithm draws $\widetilde{\boldsymbol{u}} \overset{\$}{\leftarrow} \{0;1\}^n \backslash \{\mathbf{0}_n\}$, $\widetilde{\mathsf{e}}, \widetilde{\mathsf{e}}' \overset{\$}{\leftarrow} \chi_{2^\lambda \sigma}$, and sets, with $\widetilde{\mathsf{u}} \leftarrow \mathsf{pol}_{\mathcal{R}_p}(\widetilde{\boldsymbol{u}})$:

$$(\widetilde{\mathsf{c}}, \widetilde{\mathsf{c}}') \leftarrow (\mathsf{c}, \mathsf{c}') + (\mathsf{p} \cdot \widetilde{\mathsf{u}} + \widetilde{\mathsf{e}}, \mathsf{p}' \cdot \widetilde{\mathsf{u}} + \widetilde{\mathsf{e}}').$$

It now denotes $\widetilde{\boldsymbol{c}} = (\widetilde{c_1}, \ldots, \widetilde{c_n}) \leftarrow \mathsf{pol}^{-1}_{\mathcal{R}_p}(\widetilde{\mathsf{c}})$, $\widetilde{\boldsymbol{c}}' = (\widetilde{c_1}', \ldots, \widetilde{c_n}') \leftarrow \mathsf{pol}^{-1}_{\mathcal{R}_p}(\widetilde{\mathsf{c}}')$, $\widetilde{\boldsymbol{w}} = (\widetilde{w_1}, \ldots, \widetilde{w}_{\widetilde{N}}) \in \{0;1\}^{\widetilde{N}}$ (with $\widetilde{N} \leftarrow n(1 + 2B_{\mathsf{Rand}})$) the vector of the bit decomposition of $\widetilde{\boldsymbol{u}} \| \mathsf{pol}^{-1}_{\mathcal{R}_p}(\widetilde{\mathsf{e}}) \| \mathsf{pol}^{-1}_{\mathcal{R}_p}(\widetilde{\mathsf{e}}')$.

Tracing and Validity Proofs Using the proof $\mathit{\Pi}_{\mathsf{SiSo}}$ kept in C, the algorithm processes the following steps:

1. it modifies $\mathcal{E}_{\mathsf{SiSo}}$ into $\widetilde{\mathcal{E}_{\mathsf{SiSo}}}$ encoding the following system of equations in variables $\mathfrak{X}, (Y_j)_j, (\widetilde{Z}_k)_k, (\widetilde{\mathfrak{Z}}_k)_k, (X_1, X_2), (Z_i)_i, (\mathfrak{Z}_i)_i, (X_1', X_2'), (Y_1', Y_2')$:

$$\left\{\begin{array}{ll}
e(\widetilde{c_1} G, \mathfrak{X}) = e(Y_1, \mathfrak{G}) & \\
\qquad\qquad \vdots & \\
e(\widetilde{c_n}' G, \mathfrak{X}) = e(Y_{2n}, \mathfrak{G}) & \\
e(G, \mathfrak{X}) = e(Y_{2n+1}, \mathfrak{G}) & \qquad (1a) \\
\forall k \in \left[\!\left[1; \widetilde{N} \right]\!\right] : e(\widetilde{Z}_k, \mathfrak{X}) = e(Y_{2n+1+k}, \mathfrak{G}) & \\
 & \\
e(X_1, \mathfrak{G}) e(X_2, \mathfrak{H}) = \prod_{j=1}^{2n+1+\widetilde{N}} e(Y_j, \mathfrak{G}_{\mathsf{opk},j}) & \qquad (1b) \\
 & \\
\forall j \in [\![1; n]\!] : & \\
\quad \prod_{i=1}^{N} e(Z_i, \mathfrak{X})^{p_{i,j}} \cdot \prod_{k=1}^{\widetilde{N}} e(\widetilde{Z}_k, \mathfrak{X})^{\widetilde{p}_{i,j}} = e(\widetilde{c_j} G, \mathfrak{X}) & \qquad (1c) \\
\quad \prod_{i=1}^{N} e(Z_i, \mathfrak{X})^{p_{i,j+n}} \cdot \prod_{k=1}^{\widetilde{N}} e(\widetilde{Z}_k, \mathfrak{X})^{\widetilde{p}_{i,j+n}} = e(\widetilde{c_j}' G, \mathfrak{X}) & \\
 & \\
\forall i \in [\![1; N]\!] : e(Z_i, \mathfrak{Z}_i) = e(G, \mathfrak{Z}_i) & \qquad (1d) \\
\qquad\qquad\qquad e(Z_i, \mathfrak{G}) = e(G, \mathfrak{Z}_i) & \\
 & \\
\forall k \in \left[\!\left[1; \widetilde{N} \right]\!\right] : e(\widetilde{Z}_i, \widetilde{\mathfrak{Z}}_i) = e(G, \widetilde{\mathfrak{Z}}_i) & \qquad (1e) \\
\qquad\qquad\qquad e(\widetilde{Z}_i, \mathfrak{G}) = e(G, \widetilde{\mathfrak{Z}}_i) & \\
 & \\
e(X_1', \mathfrak{G}) e(X_2', \mathfrak{H}) = e(Y_1', \mathfrak{G}_{\mathsf{spk},1}) e(Y_2', \mathfrak{G}_{\mathsf{spk},2}) & \\
e(G, \mathfrak{X}) e(Y_1', \mathfrak{G}) = e(G, \mathfrak{G}) & \qquad (1f) \\
e(\tau G, \mathfrak{X}) e(Y_2', \mathfrak{G}) = e(\tau G, \mathfrak{G}) &
\end{array}\right.$$

2. and then computes of proof of knowledge of a solution in the following manner:
   – to obtain proofs and commitments for equations 1a and 1b, it computes:

$$\begin{aligned}
\forall j \in [\![1; 2n]\!] : \mathsf{Com}_{\mathfrak{B}, \widetilde{\boldsymbol{T}}, j} &\leftarrow \sum_{i=1}^{\widetilde{N}} \widetilde{w_i} \cdot \mathsf{Com}_{\boldsymbol{T}_i, j} \\
\mathsf{Com}_{\mathfrak{B}, \widetilde{\boldsymbol{T}}, 2n+1} &\leftarrow \mathsf{Com}_{\mathfrak{B}, G} \\
\forall j \in \left[\!\left[2n+2; 2n+1+\widetilde{N} \right]\!\right] : \mathsf{Com}_{\mathfrak{B}, \widetilde{\boldsymbol{T}}, j} &\leftarrow \widetilde{w}_{j-2n-1} \cdot \mathsf{Com}_{\mathfrak{B}, G} \\
\mathsf{Com}_{\sigma} &\leftarrow \mathsf{Com}_{\sigma,1} + \sum_{i=1}^{\widetilde{N}} \widetilde{w_i} \cdot \mathsf{Com}_{\sigma, i+1} \\
\forall j \in [\![1; 2n]\!] : \pi_{\mathfrak{B}, j} &\leftarrow \pi_{\mathfrak{B},1,j} + \sum_{i=1}^{\widetilde{N}} \widetilde{w_i} \cdot \pi_{\mathfrak{B}, i+1, j} \\
\pi_{\mathfrak{B}, 2n+1} &\leftarrow \pi_{\mathfrak{B},1} \\
\forall k \in \left[\!\left[1; \widetilde{N} \right]\!\right] : \pi_{\mathfrak{B}, 2n+1+k} &\leftarrow \widetilde{w}_k \cdot \pi_{\mathfrak{B},1} \\
\pi_{\mathsf{Sign},\mathsf{opk}} &\leftarrow \pi_{\mathsf{Sign},1} + \sum_{i=1}^{\widetilde{N}} \widetilde{w_i} \cdot \pi_{\mathsf{Sign}, i+1}
\end{aligned}$$

– a proof for equations 1c is also obtained using $(\pi_{\mathsf{IP},j})_j$, $\mathsf{Com}_{\mathfrak{B}}$ and $\mathsf{Com}_{\boldsymbol{W}}$, along with the new $\widetilde{\boldsymbol{w}}$ vector, using the homomorphism of Groth-Sahai proofs; the algorithm commits to $\widetilde{\boldsymbol{w}}G$ with null randomness, computing: $\mathsf{Com}_{\widetilde{\boldsymbol{W}}} \leftarrow \widetilde{\boldsymbol{w}}G \cdot (0,1)$ and setting $\mathsf{Com}_{\boldsymbol{W}\|\widetilde{\boldsymbol{W}}} \leftarrow \mathsf{Com}_{\boldsymbol{W}}\|\mathsf{Com}_{\widetilde{\boldsymbol{W}}}$; $\left(\begin{pmatrix}\mathfrak{D} & \mathfrak{D}\\ \mathfrak{D} & \mathfrak{D}\end{pmatrix}, \begin{pmatrix}O & O\\ O & O\end{pmatrix}\right)$ is then a trivial proof of knowledge of a solution to:

$$\begin{cases} \prod_{i=1}^{\widetilde{N}} e(\widetilde{Z}_i, \mathfrak{X})^{\widetilde{p}_{i,j}} & = e((\widetilde{c}_j - c_j)G, \mathfrak{X}) \\ \prod_{i=1}^{\widetilde{N}} e(\widetilde{Z}_i, \mathfrak{X})^{\widetilde{p}_{i,j+n}} & = e((\widetilde{c}_j{}' - c_j')G, \mathfrak{X}) \end{cases}$$

with respect to $\mathsf{Com}_{\mathfrak{B}}$ and $\mathsf{Com}_{\widetilde{\boldsymbol{W}}}$, and thus $(\pi_{\mathsf{IP},\mathsf{rand},j})_j \leftarrow (\pi_{\mathsf{IP},j})_j$ yields a proof of equations 1c with respect to $\mathsf{Com}_{\mathfrak{B}}$ and $\mathsf{Com}_{\boldsymbol{W}\|\widetilde{\boldsymbol{W}}}$;

– a proof of the equations 1d is already provided by $(\pi_{\mathsf{Bit},k})_{k\in[\![1;N]\!]}$, which is renamed as: $(\pi_{\mathsf{Bit},\mathsf{fresh},k})_{k\in[\![1;N]\!]}$;

– to build a proof of equations 1e, the algorithm uses $\mathsf{Com}_{\widetilde{\boldsymbol{W}}}$, along with a new commitment $\mathsf{Com}_{\widetilde{\boldsymbol{\mathfrak{W}}}}$ to $\widetilde{\boldsymbol{w}} \cdot \mathfrak{G}$ (yielding $\mathsf{Com}_{\boldsymbol{\mathfrak{W}}\|\widetilde{\boldsymbol{\mathfrak{W}}}} \leftarrow \mathsf{Com}_{\boldsymbol{\mathfrak{W}}}\|\mathsf{Com}_{\widetilde{\boldsymbol{\mathfrak{W}}}}$), to build corresponding proofs $(\pi_{\mathsf{Bit},\mathsf{Rand},k})_{k\in[\![1;\widetilde{N}]\!]}$ using $\mathsf{Com}_{\widetilde{\boldsymbol{W}}}$'s known randomness;

– finally, a proof of knowledge of a solution to equations 1f is already provided by π_τ, with respect to $\mathsf{Com}_{\mathfrak{B}}$, $\mathsf{Com}_{\sigma,\tau}$ and Com_τ.

3. the algorithm then randomizes all the above proofs and commitments in the Groth-Sahai framework, which yields:

$$\widetilde{\mathsf{Com}} \leftarrow \widetilde{\mathsf{Com}_{\mathfrak{B}}}\|\widetilde{\mathsf{Com}_{\sigma}}\|\widetilde{\mathsf{Com}_{\mathfrak{B},\widetilde{\boldsymbol{T}}}}\|\widetilde{\mathsf{Com}_{\boldsymbol{W}\|\widetilde{\boldsymbol{W}}}}\|\widetilde{\mathsf{Com}_{\boldsymbol{\mathfrak{W}}\|\widetilde{\boldsymbol{\mathfrak{W}}}}}\|\widetilde{\mathsf{Com}_{\sigma,\tau}}\|\widetilde{\mathsf{Com}_{\tau}}$$
$$\widetilde{\pi} \leftarrow (\widetilde{\pi}_{\mathfrak{B},j})_j\|\widetilde{\pi}_{\mathsf{Sign},\mathsf{opk}}\|(\widetilde{\pi}_{\mathsf{IP},\mathsf{fresh},j})_j\|(\widetilde{\pi}_{\mathsf{IP},\mathsf{rand},j})_j\|(\widetilde{\pi}_{\mathsf{Bit},\mathsf{fresh},i})_i\|(\widetilde{\pi}_{\mathsf{Bit},\mathsf{Rand},i})_i\|\widetilde{\pi}_\tau$$

as a proof of knowledge $\widetilde{\Pi}_{\mathsf{SiSo}} \leftarrow (\widetilde{\mathsf{Com}}, \widetilde{\pi}, \widetilde{\mathcal{E}}_{\mathsf{SiSo}})$ with fresh randomness.

Output: the algorithm sets the TREnc ciphertext to: $\widetilde{C} \leftarrow ((\widetilde{\mathsf{c}}, \widetilde{\mathsf{c}}'), \mathsf{opk}, \widetilde{\Pi}_{\mathsf{SiSo}})$, and returns it.

4.5 Verification Algorithm Vf

Input: PK parsed as a such-named output of Gen, C as a corresponding output of Enc or Rand, and a label $\ell \in \{\mathsf{fresh}, \mathsf{rand}\}$ indicating whether it should be validated as an output of Enc or of Rand.

Computations and Output: First, the algorithm verifies that the system of equations $\mathcal{E}_{\mathsf{SiSo}}$ described in Π_{SiSo} indeed corresponds to $(\mathsf{c}, \mathsf{c}')$, opk (by verifying that $\tau \leftarrow \mathcal{H}(\mathsf{opk}, \mathsf{PK})$ was correctly derived), and the public vector values $\boldsymbol{p}, \widetilde{\boldsymbol{p}}$, and public parameters pp; if not, it sets and returns $b \leftarrow 0$. Then, it checks whether: $\mathsf{GS.Vf}(\mathsf{pp}, \varphi, \Pi_{\mathsf{SiSo}}) = 1$, and if not, sets and returns $b \leftarrow 0$.

4.6 Decryption Algorithm Dec

Taking as input sk and PK parsed as corresponding outputs of Gen, and C parsed as an output of Enc or Rand, this algorithm returns: $\perp$ if $\mathsf{Vf}(C, \mathsf{PK}, \mathsf{fresh}) = \mathsf{Vf}(C, \mathsf{PK}, \mathsf{rand}) = 0$; and else: $\mathsf{m} \leftarrow \mathsf{PQPKE.Dec}(\mathsf{sk}, (\mathsf{c}, \mathsf{c}'))$.

5 Security of the Protocol

The $\mathsf{TREnc} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Trace}, \mathsf{Rand}, \mathsf{Dec}, \mathsf{Vf})$ scheme described in Section 4 is showed to verify security properties of Traceable Receipt-free Encryption (TREnc) schemes.

Theorem 11 (Correctness of TREnc). TREnc *is correct under the correctness of* PQPKE*,* GS *proofs, and* LHSP *signatures.*

Proof. Straightforward.

Theorem 12 (Verifiability of TREnc). TREnc *is verifiable under the SXDH assumption. More precisely, for any PPT adversary $\mathcal{A}$ and security parameter $\lambda \in \mathbb{N}$,* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Ver},\mathsf{TREnc}}(\lambda) \leqslant 6 \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathsf{SXDH}}(\lambda)$.

Proof. See Appendix B.1.

Theorem 13 (Traceability of TREnc). TREnc *is traceable under the SXDH assumption. More precisely, for any security parameter $\lambda \in \mathbb{N}$ and PPT adversary $\mathcal{A}$:* $\Pr\{\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Trace}}(\lambda) = 1\} \leqslant 2^{-2n} + \frac{1}{p} + 4 \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathsf{SXDH}}(\lambda)$.

Proof. See Appendix B.2.

Theorem 14 (TCCA Security of TREnc). TREnc *is TCCA-secure under the SXDH and RLWE assumptions and the security of the hash function against collisions. More precisely, for any PPT adversary $\mathcal{A}$ and security parameter $\lambda \in \mathbb{N}$:* $\mathsf{Adv}_{\mathcal{A}}^{TCCA}(\lambda) \leqslant \mathsf{Adv}_{\mathcal{A}}^{\mathsf{coll},\mathcal{H}}(\lambda) + \frac{1}{p} + 3 \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathsf{SXDH}}(\lambda) + \mathsf{Adv}_{\mathcal{A}}^{\mathsf{RLWE}}(\lambda)$.

Proof. See Appendix B.3.

Theorem 15 (wCCA Security of TREnc). TREnc *is wCCA-secure under the SXDH and RLWE assumptions and the security of the hash function against collisions. More precisely, for any PPT adversary $\mathcal{A}$ and security parameter $\lambda \in \mathbb{N}$:* $\mathsf{Adv}_{\mathcal{A}}^{wCCA}(\lambda) \leqslant \mathsf{Adv}_{\mathcal{A}}^{\mathsf{coll},\mathcal{H}}(\lambda) + \frac{1}{p} + 3 \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathsf{SXDH}}(\lambda) + \mathsf{Adv}_{\mathcal{A}}^{\mathsf{RLWE}}(\lambda)$.

Proof. Obtained straighforwardly following the TCCA proof steps (the only difference being that in the challenge phase, the adversary now sends cleartexts, and gets a fresh ciphertext as an answer), as ciphertexts before and after the randomization have the same FV structure.

Theorem 16 (Post-Quantum IND-CPA Security of TREnc). TREnc *is IND-CPA-secure under the RLWE assumption; for any PPT adversary $\mathcal{A}$,* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{CPA},\mathsf{TREnc}}(\lambda) \leqslant \mathsf{Adv}_{\mathcal{A}}^{\mathsf{RLWE}}(\lambda)$.

Proof. See Appendix B.4.

6 Applications

Efficiency Public-keys consist of 5 elements of $\mathbb{G}$, 7 elements of $\hat{\mathbb{G}}$, and a public key of the post-quantum LWE-based scheme, given by $2n$ elements of $\mathbb{Z}_p$ in the case of the FV instantiation; according to the [APS15] estimator, with p on 129 bits, and a standard deviation $\sigma \leftarrow 1$, $n \leftarrow 2^{12}$ will provide an LWE security on more than 142 bits (so a reasonable RLWE security); taking $\mathbb{G}$ elements on 310 bits and $\hat{\mathbb{G}}$ elements on 620 bits will then lead to public keys on 128 KB.

A fresh ciphertext consists of $32n + 6B_{\mathsf{fresh}} + 10N + 2\widetilde{N} + 26 = 54n + \lfloor \log_2(B) \rfloor (6 + 24n) + 26$ elements of $\mathbb{G}$, $10N + 2\widetilde{N} + 26n + 4B_{\mathsf{fresh}} + 19 = 48n + (24n + 4)\lfloor \log_2(B) \rfloor + 19$ elements of $\hat{\mathbb{G}}$, and one LWE-based ciphertext, consisting of $2n$ elements of $\mathbb{Z}_p$ in the case of the FV instantiation; for a security on 128 bits, this will represent, with the selected parameters and $B = 10\sigma$, around 55.5MB.

A randomized ciphertext consists of $26 + 20n + 16\widetilde{N} + 10N \leqslant 26 + 2n(10 + 26\lfloor \log_2(B) \rfloor)$ elements of $\mathbb{G}$, $19 + 10N + 14\widetilde{N} + 18n \leqslant 19 + 2n(9 + 24\lfloor \log_2(B) \rfloor)$ elements of $\hat{\mathbb{G}}$, and one LWE-based ciphertext, consisting of $2n$ elements of $\mathbb{Z}_p$ in the case of the FV instantiation; for a security on 128 bits, this will represent, with the selected parameters, around 69.1MB.

Receipt-Free & Ballot-Private EVoting The design of the TREnc primitive in [DPP22] was motivated by capturing simple yet sufficient conditions of a verifiable public-key encryption that naturally yields a voting system with a non-interactive voting process that offers ballot privacy and receipt freeness. The main definitional novelty to generically build a receipt-free voting system lied both in the ability for the users/voters to trace their encrypted messages/votes when their ciphertexts/ballots appear on a bulletin board after being randomized/processed by a randomizing server while being sure that their content has been altered even by the authorities (traceability), and in the privacy notion defined for the first time as an indistinguishability notion achieved by randomization (TCCA). Even if the randomizing server is deemed honest to provide the receipt-freeness by honestly randomizing valid ciphertext before publishing them on a bulletin board, the server is considered malicious when it comes to prove the ballot privacy. Roughly speaking, this notion is satisfied if no efficient adversary is able to distinguish whether honest ballots are compatible with the result of the election [BCG+15]. Although the trust model differs for receipt freeness and ballot privacy, a voting system based on a TCCA-secure TREnc that also enjoys a strong randomization notion [DPP22] is naturally ballot private.

In our more general definition of TREnc, it is straightforward to see that the adaptive-trace weak CCA notion given in Defintion 10 is sufficient to imply the ballot privacy of a voting system that encrypts votes with a TREnc. Moreover, our TCCA definition is still equivalent of the original definition as long as the chosen ciphertexts are valid for the fresh ciphertext space $\mathcal{C}_{\mathsf{fresh}}$. Since a verifiable TREnc allows identifying those ciphertexts and since the generic voting system of [DPP22] defines the voting algorithm essentially as the encryption of the encoded-vote message, we keep the receipt freeness.

6.1 Voting System Security Notions

The generic transformation of our TREnc construction into a voting scheme follows the same recipy as the one of the original paper [DPP22]; we recall the corresponding definitions and security notions here.

Definition 17 (Voting System (from [DPP22])). *A Voting System is a tuple of probabilistic polynomial-time algorithms (*$\mathsf{SetupElection}$*,* Vote*,* $\mathsf{ProcessBallot}$*,* $\mathsf{TraceBallot}$*,* Valid*,* Append*,* $\mathsf{Publish}$*,* $\mathsf{VerifyVote}$*,* Tally*,* $\mathsf{VerifyResult}$*) associated to a result function* $\rho_m : \mathcal{V}^m \cup \perp \to \mathcal{R}$ *where* $\mathcal{V}$ *is the set of valid votes and* $\mathcal{R}$ *is the result space such that:*

$\mathsf{SetupElection}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$**:** *on input a security parameter* λ*, generates the public and secret key pair* $(\mathsf{pk}, \mathsf{sk})$ *of the election.*

$\mathsf{Vote}(\mathsf{id}, v) \to (\mathsf{b}, \mathsf{aux})$**:** *when receiving a voter* id *and a vote* v*, outputs a ballot* b *and auxiliary data* aux*. It will also be possible to call* $\mathsf{Vote}(\mathsf{id}, v, \mathsf{aux})$ *in order to obtain a ballot (without auxiliary data this time) for the vote* v *using* aux*. This auxiliary data will be useful to define security and enables the creation of ballots that share the same* aux*.*

$\mathsf{ProcessBallot}(\mathsf{b}) \to \widetilde{\mathsf{b}}$**:** *on input a ballot* b*, outputs an updated ballot* $\widetilde{\mathsf{b}}$*. In our case,* $\widetilde{\mathsf{b}}$ *will be a rerandomization of* b*.*

$\mathsf{TraceBallot}(\mathsf{b}) \to \tau$**:** *on input a ballot* b*, outputs a tag* τ*. The tag is the information that a voter can use to trace her ballot, using the* $\mathsf{VerifyVote}$ *algorithm.*

$\mathsf{Valid}(\mathsf{BB}, \mathsf{b}) \to b$**:** *on input a ballot box* BB *and ballot* b*, outputs 1 if and only if the ballot is valid, and else 0.*

$\mathsf{Append}(\mathsf{BB}, \mathsf{b}) \to \widetilde{\mathsf{BB}}$**:** *on input a ballot box* BB *and ballot* b*, appends* $\mathsf{ProcessBallot}(\mathsf{b})$ *to* BB *iff* $\mathsf{Valid}(\mathsf{BB}, \mathsf{b}) = 1$*, and then returns the updated (or identical) ballot box* $\widetilde{\mathsf{BB}}$*.*

$\mathsf{Publish}(\mathsf{BB}) \to \mathsf{PBB}$**:** *on input a ballot box* BB*, outputs the public view* PBB *of* BB*, which is the one that is used to verify the election. Depending on the context, it may for instance be used to remove some voter credentials.*

$\mathsf{VerifyVote}(\mathsf{PBB}, \tau) \to b$**:** *on input a public ballot box* PBB *and tag* τ*, outputs a bit* b *equal to 1 iff the vote corresponding to the tag* τ *(which is specific to a voter) has been processed and recorded properly.*

$\mathsf{Tally}(\mathsf{BB}, \mathsf{sk}) \to (\mathsf{r}, \Pi)$**:** *on input a ballot box* BB *and the election secret key* sk*, outputs the tally* r *and a proof* Π *that the tally is correct with respect to the result function* ρ_m*.*

$\mathsf{VerifyResult}(\mathsf{PBB}, \mathsf{r}, \Pi) \to b$**:** *on input a public ballot box* PBB*, tally result* r *and tally proof* Π*, outputs a bit* b *equal to 1 if and only if* Π *is a valid proof that* r *is the election result, computed with respect to* ρ_m*, corresponding to the ballots on* PBB*.*

For all of these algorithms except $\mathsf{SetupElection}$*, the public key of the election* pk *is an implicit argument.*

A voting system should follow the following security notions:

Definition 18 (Tracing Correctness (from [DPP22])). *A voting system verifies tracing correctness iff for* $\lambda \in \mathbb{N}$, $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{SetupElection}(1^\lambda)$*, and every* $v, \mathsf{BB}, (\mathsf{b}, \mathsf{aux}) \xleftarrow{\$} \mathsf{Vote}(\mathsf{id}, v)$ *and* $\tau \leftarrow \mathsf{TraceBallot}(\mathsf{b})$*,for* $\widetilde{\mathsf{BB}} \leftarrow \mathsf{Append}(\mathsf{BB}, \mathsf{b})$, $\mathsf{VerifyVote}(\mathsf{Publish}(\mathsf{BB}), \tau) = 1$ *with overwhelming probability in* λ*.*

Definition 19 (Receipt-Freeness (from [DPP22])). *A voting system* V *verifies receipt-freeness iff there exist algorithms* $\mathsf{SimSetupElection}$ *and* $\mathsf{SimProof}$ *such that, for* $\lambda \in \mathbb{N}$*, any PPT adversary* $\mathcal{A}$*'s advantage in distinguishing the games* $\mathsf{Exp}^{\mathsf{RF},0}_{\mathcal{A},V}(\lambda)$ *and* $\mathsf{Exp}^{\mathsf{RF},1}_{\mathcal{A},V}(\lambda)$ *defined by the oracles in figure 6.1 is negligible in* λ*.*

```
$\mathcal{O}\mathsf{init}^\beta(\lambda)$:
---------------------------------------
**if** $\beta = 0$ **then**
    $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{SetupElection}(1^\lambda)$
**else** $(\mathsf{pk}, \mathsf{sk}, \tau) \xleftarrow{\$} \mathsf{SimSetupElection}(1^\lambda)$

$\mathsf{BB}_0 \leftarrow \varnothing, \mathsf{BB}_1 \leftarrow \varnothing$
**return** $\mathsf{pk}$.
```

```
$\mathcal{O}\mathsf{receiptLR}(\mathsf{b}_0, \mathsf{b}_1)$:
---------------------------------------
**if** $\mathsf{TraceBallot}(\mathsf{b}_0) \neq \mathsf{TraceBallot}(\mathsf{b}_1)$
    or $\mathsf{Valid}(\mathsf{BB}_0, \mathsf{b}_0) = 0$
    or $\mathsf{Valid}(\mathsf{BB}_1, \mathsf{b}_1) = 0$
    **then** return $\perp$
**else** $\mathsf{BB}_0 \leftarrow \mathsf{Append}(\mathsf{BB}_0, \mathsf{b}_0)$,
    $\mathsf{BB}_0 \leftarrow \mathsf{Append}(\mathsf{BB}_0, \mathsf{b}_0)$.
```

```
$\mathcal{O}\mathsf{board}^\beta$:
---------------------------------------
**return** $\mathsf{Publish}(\mathsf{BB}_\beta)$.
```

```
$\mathcal{O}\mathsf{tally}^\beta$:
---------------------------------------
$(\mathsf{r}, \Pi) \xleftarrow{\$} \mathsf{Tally}(\mathsf{BB}_0, \mathsf{sk})$
**if** $\beta = 1$ **then** $\Pi \xleftarrow{\$} \mathsf{SimProof}(\mathsf{BB}_1, \mathsf{r}, \tau)$
**return** $(\mathsf{r}, \Pi)$.
```

Fig. 4. Oracles used in the $\mathsf{Exp}^{\mathsf{RF},\beta}_{\mathcal{A},V}(\lambda)$ experiment, for $\beta \in \{0; 1\}$. The adversary first calls $\mathcal{O}\mathsf{init}^\beta(\lambda)$, and may then call the $\mathcal{O}\mathsf{board}$ and $\mathcal{O}\mathsf{receiptLR}$ oracles as much as she wants. She finally cacls $\mathcal{O}\mathsf{tally}$, receives the result of the election, and is requested to output her guess β' for the value of β, which is the output of the experiment.

Definition 20 (Ballot Traceability for Receipt-Freeness (from [DPP22])). *For every* pk *in the range of* $\mathsf{SetupElection}$*, voter identity* id*, and pair of votes* v_0, v_1*, for* $(\mathsf{b}_0, \mathsf{aux}) \xleftarrow{\$} \mathsf{Vote}(\mathsf{id}, v_0)$ *and* $\mathsf{b}_1 \xleftarrow{\$} \mathsf{Vote}(\mathsf{id}, v_1, \mathsf{aux})$, $\mathsf{TraceBallot}(\mathsf{b}_0) = \mathsf{TraceBallot}(\mathsf{b}_1)$.

6.2 Voting System Security Proofs

Deriving the voting system resulting from the above TREnc as in [DPP22] also yields a secure system with the same voting system security notions, even with our more general TREnc ones. Independently of the TREnc scheme, the derivation of the voting scheme requires the use of a scheme computing the result of the election from the ciphertexts (this may be done using mixnets, homomorphic computations, or a more general MPC procedure), along with proofs that this result was calculated correctly. As the choice of this building block is modular with respect to our TREnc construction, we leave it to the implementor;

however, the receipt-freeness, along with relying on the TCCA security of the TREnc scheme, will rely on the zero-knowledge property of the proving scheme used for the tally calculation of an election result, providing a $\mathsf{SimSetupElection}$ algorithm with $(\mathsf{pk}, \mathsf{sk})$ outputs indistinguishable from the $\mathsf{SetupElection}$ one, and a $\mathsf{SimProof}$ algorithm with outputs indistinguishable from those of Tally for an adversary with pk. $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{SimSetup}}\mathsf{Election}(\lambda)$ will denote an adversary $\mathcal{A}$'s advantage in distinguishing outputs $(\mathsf{pk}, \mathsf{sk})$ from $\mathsf{SimSetupElection}$ or $\mathsf{SetupElection}$, and $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{SimProof}}(\lambda)$ in distinguishing an output Π from $\mathsf{SimProof}$ or Tally, given the corresponding result r and public key pk, for a security parameter λ.

Theorem 21 (Receipt-Freeness). *If a TREnc scheme is TCCA and the tally result is proven using a zero-knowledge scheme yielding indistinguishable algorithms* $\mathsf{SimSetupElection}$ *and* $\mathsf{SetupElection}$*, and* Tally *and* $\mathsf{SimProof}$*, then the corresponding voting system is receipt-free; more precisely, for any PPT adversary* $\mathcal{A}$*,* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{RF}}(\lambda) \leqslant Q_{\mathcal{O}}\mathsf{receiptLR} \cdot \mathsf{Adv}_{\mathcal{A}}^{TCCA}(\lambda) + Q_{\mathcal{O}\mathsf{tally}} \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathsf{SimProof}}(\lambda) + \mathsf{Adv}_{\mathcal{A}}^{\mathsf{SimSetupElection}}(\lambda)$*, for* $Q_{\mathcal{O}}\mathsf{receiptLR}$ *the number of requests* $\mathcal{A}$ *makes to the* $\mathcal{O}\mathsf{receiptLR}$ *oracle, and* $\mathcal{O}\mathsf{tally}$ *the number of requests she sends to* $\mathcal{O}\mathsf{tally}$*.*

Proof. See Appendix B.5.

Moreover, the traceability of TREnc immediately yields the ballot traceability of the voting scheme. As for Ballot Privacy and Verifiability, they naturally follow as in the [DPP22] proofs.

References

ACPS09. Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 595–618. Springer, Heidelberg, August 2009.

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

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

APS15. Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.

BCG+15. David Bernhard, Véronique Cortier, David Galindo, Olivier Pereira, and Bogdan Warinschi. Sok: A comprehensive analysis of game-based ballot privacy definitions. *2015 IEEE Symposium on Security and Privacy*, pages 499–516, 2015.

BdPP23. Théophile Brézot, Paola de Perthuis, and David Pointcheval. Covercrypt: an efficient early-abort kem for hidden access policies with traceability from the ddh and lwe. LNCS, pages 372–392. Springer, Heidelberg, 2023.

BF11. Dan Boneh and David Mandell Freeman. Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 1–16. Springer, Heidelberg, March 2011.

BS22. Ward Beullens and Gregor Seiler. LaBRADOR: Compact proofs for R1CS from module-SIS. Cryptology ePrint Archive, Report 2022/1341, 2022. https://eprint.iacr.org/2022/1341.

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.

CKLM12. Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Malleable proof systems and applications. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 281–300. Springer, Heidelberg, April 2012.

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.

DPP24. Henri Devillez, Olivier Pereira, and Thomas Peters. Practical traceable receipt-free encryption. In Clemente Galdi and Duong Hieu Phan, editors, *Security and Cryptography for Networks - 14th International Conference, SCN 2024, Amalfi, Italy, September 11-13, 2024, Proceedings, Part I*, volume 14973 of *Lecture Notes in Computer Science*, pages 367–387. Springer, 2024.

FV12. Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. https://eprint.iacr.org/2012/144.

Gro06. Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 444–459. Springer, Heidelberg, December 2006.

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.

GVW15. Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. Leveled fully homomorphic signatures from standard lattices. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 469–477. ACM Press, June 2015.

LNP22. Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plançon. Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 71–101. Springer, Heidelberg, August 2022.

LPJY14. Benoît Libert, Thomas Peters, Marc Joye, and Moti Yung. Non-malleability from malleability: Simulation-sound quasi-adaptive NIZK proofs and CCA2-secure encryption from homomorphic signatures. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 514–532. Springer, Heidelberg, May 2014.

LPR10. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23. Springer, Heidelberg, May / June 2010.

MRY04. Philip D. MacKenzie, Michael K. Reiter, and Ke Yang. Alternatives to non-malleability: Definitions, constructions, and applications (extended abstract). In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 171–190. Springer, Heidelberg, February 2004.

Ràf15. Carla Ràfols. Stretching groth-sahai: NIZK proofs of partial satisfiability. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 247–276. Springer, Heidelberg, March 2015.

Appendix

A The Groth-Sahai Proof System [GS08]

A Groth-Sahai (GS) proof system is defined with the following algorithms for proofs on group elements:

$\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$**:** on input the security parameter $\lambda \in \mathbb{N}$, returns public parameters pp providing a pairing setting with: $\mathsf{pp} \leftarrow (p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e, G, \mathfrak{G})$;

$\mathsf{BCRSGen}(\mathsf{pp}) \to \mathsf{crs}$**:** on input pp parsed as an output of Setup, generates a common reference string in the perfectly binding mode; the algorithm draws $a, t, \mathfrak{a}, \mathfrak{t} \xleftarrow{\$} \mathbb{Z}_p$, and sets: $\boldsymbol{U}_1 \leftarrow (G, aG), \boldsymbol{U}_2 \leftarrow (tG, taG), \boldsymbol{\mathfrak{U}}_1 \leftarrow (\mathfrak{G}, \mathfrak{a}\mathfrak{G}), \boldsymbol{\mathfrak{U}}_2 \leftarrow (\mathfrak{t}\mathfrak{G}, \mathfrak{t}\mathfrak{a}\mathfrak{G})$, finally returning: $\mathsf{crs} \leftarrow (\boldsymbol{U}_1, \boldsymbol{U}_2, \boldsymbol{\mathfrak{U}}_1, \boldsymbol{\mathfrak{U}}_2)$.

$\mathsf{HCRSGen}(\mathsf{pp}) \to \mathsf{crs}$**:** on input pp parsed as an output of Setup, generates (except with negligible probability in λ) a common reference string in the perfectly witness-indistinguishable (hiding) mode, by setting and returning: $\mathsf{crs} \leftarrow (\boldsymbol{U}_1, \boldsymbol{U}_2, \boldsymbol{\mathfrak{U}}_1, \boldsymbol{\mathfrak{U}}_2) \xleftarrow{\$} \mathbb{G}^2 \times \hat{\mathbb{G}}^2$.

$\mathsf{Com\&Pr}(\mathsf{pp}, \mathsf{crs}, w, \mathcal{E}) \to \pi$**:** for the set of equations $\mathcal{E}$ (each one of them defined by equation-specific vectors $\boldsymbol{A}, \boldsymbol{\mathfrak{A}}$, matrices $\underline{\Gamma}$ and resulting elements T_T as defined afterwards), and a witness w listing: $\boldsymbol{X} \in \mathbb{G}^n, \boldsymbol{\mathfrak{X}} \in \hat{\mathbb{G}}^k$, that will verify all the equations at once, this algorithm will enable the proof of the set of pairing-product equations defined in $\mathcal{E}$, of the form $\langle \boldsymbol{A}; \boldsymbol{\mathfrak{X}} \rangle \langle \boldsymbol{X}; \boldsymbol{\mathfrak{A}} \rangle \boldsymbol{X}^T \underline{\Gamma} \boldsymbol{\mathfrak{X}} = T_T$, for $\boldsymbol{A} \in \mathbb{G}^k, \boldsymbol{\mathfrak{A}} \in \hat{\mathbb{G}}^n, \underline{\Gamma} \in \mathbb{Z}_p^{n \times k}, T_T \in \mathbb{G}_T$, in the variables $\boldsymbol{\mathfrak{X}} \in \hat{\mathbb{G}}^k, \boldsymbol{X} \in \mathbb{G}^n$, where the multiplication operation between an element of $\mathbb{G}$ and an element $\hat{\mathbb{G}}$ is the pairing operation e, and the addition operation in $\mathbb{G}_T$ is actually a multiplication, as $\mathbb{G}_T$ is a multiplicative group: *i. e.*, with $\boldsymbol{A} = (A_1, \ldots, A_k) \in \mathbb{G}^k, \boldsymbol{\mathfrak{X}} = (\mathfrak{X}_1, \ldots, \mathfrak{X}_k) \in \hat{\mathbb{G}}^k$, $\langle \boldsymbol{A}; \boldsymbol{\mathfrak{X}} \rangle = \prod_{i=1}^k e(A_i, X_i) \in \mathbb{G}_T$.

1. To prove that $(\boldsymbol{X}, \boldsymbol{\mathfrak{X}}, \boldsymbol{x}, \boldsymbol{\mathfrak{x}})$ is a valid witness for the set of equations, the algorithm first commits to each one of its elements:
   - to commit to $\boldsymbol{X} \in \mathbb{G}^n$, it draws $\underline{R} \in \mathbb{Z}_p^{n \times 2}$ and sets: $\mathbf{Com}_{\boldsymbol{X}} \leftarrow \boldsymbol{X} \cdot (0, 1) + \underline{R} \begin{pmatrix} \boldsymbol{U}_1^T \\ \boldsymbol{U}_2^T \end{pmatrix} \in \mathbb{G}^{n \times 2}$;
   - similarly, to commit to $\mathfrak{X}_i \in \hat{\mathbb{G}}^k$, it draws $\underline{\mathfrak{R}} \in \mathbb{Z}_p^{k \times 2}$ and sets: $\mathbf{Com}_{\mathfrak{X}} \leftarrow \boldsymbol{\mathfrak{X}} \cdot (0, 1) + \underline{\mathfrak{R}} \begin{pmatrix} \boldsymbol{\mathfrak{U}}_1^T \\ \boldsymbol{\mathfrak{U}}_2^T \end{pmatrix} \in \hat{\mathbb{G}}^{k \times 2}$;
2. then, the algorithm generates proofs for each of the equations to be proven, that will be stored in a proof list $\boldsymbol{\Pi}$ of length the number of equations in $\mathcal{E}$; for each pairing product equation of index ℓ in the set of equations, defined by $\boldsymbol{A}_\ell \in \mathbb{G}^k, \boldsymbol{\mathfrak{A}}_\ell \in \hat{\mathbb{G}}^n, \underline{\Gamma}_\ell \in \mathbb{Z}_p^{n \times k}$, and $T_{T,\ell} \in \mathbb{G}_T$, it draws $\underline{T}_\ell \xleftarrow{\$} \mathbb{Z}_p^{2 \times 2}$ and sets:

$$\underline{\pi}_\ell \leftarrow \underline{R}^T \cdot (\boldsymbol{\mathfrak{A}}_\ell \cdot (0,1)) + \underline{R}^T \underline{\Gamma}_\ell \cdot (\boldsymbol{\mathfrak{X}} \cdot (0,1)) + (\underline{R}^T \underline{\Gamma}_\ell \underline{\mathfrak{R}} - \underline{T}_\ell^T) \begin{pmatrix} \boldsymbol{\mathfrak{U}}_1^T \\ \boldsymbol{\mathfrak{U}}_2^T \end{pmatrix} \in \hat{\mathbb{G}}^{2 \times 2}$$

$$\underline{\theta}_\ell \leftarrow \underline{\mathfrak{R}}^T \cdot (\boldsymbol{A}_\ell \cdot (0,1)) + \underline{\mathfrak{R}}^T \underline{\Gamma}_\ell^T \cdot (\boldsymbol{X} \cdot (0,1)) + \underline{T}_\ell \begin{pmatrix} \boldsymbol{U}_1^T \\ \boldsymbol{U}_2^T \end{pmatrix} \in \mathbb{G}^{2\times 2}$$

then places $(\underline{\pi}_\ell, \underline{\theta}_\ell)$ in the ℓ-th coordinate of $\boldsymbol{\Pi}$;
As shown in [GS08], in the particular case where the equations are linear, they can be shown using only group elements (for more detail, see [GS08]).

3. Finally, the algorithm returns: $\pi \leftarrow (\mathbf{Com}_{\boldsymbol{X}} \| \mathbf{Com}_{\boldsymbol{\mathfrak{X}}}, \boldsymbol{\Pi}, \mathcal{E})$.

$\mathsf{Rand}(\mathsf{pp}, \mathsf{crs}, \pi) \rightarrow \widetilde{\pi}$**:** parsing π as an output of $\mathsf{Com\&Pr}$, this algorithm:

– picks $\underline{\widetilde{R}} \xleftarrow{\$} \mathbb{Z}_p^{n\times 2}$ and sets: $\widetilde{\mathbf{Com}}_{\boldsymbol{X}} \leftarrow \mathbf{Com}_{\boldsymbol{X}} + \underline{\widetilde{R}} \begin{pmatrix} \boldsymbol{U}_1^T \\ \boldsymbol{U}_2^T \end{pmatrix} \in \mathbb{G}^{n\times 2}$;

– picks $\underline{\widetilde{\mathfrak{R}}} \xleftarrow{\$} \mathbb{Z}_p^{k\times 2}$ and sets: $\widetilde{\mathbf{Com}}_{\boldsymbol{\mathfrak{X}}} \leftarrow \mathbf{Com}_{\boldsymbol{\mathfrak{X}}} + \underline{\widetilde{\mathfrak{R}}} \begin{pmatrix} \boldsymbol{\mathfrak{U}}_1^T \\ \boldsymbol{\mathfrak{U}}_2^T \end{pmatrix} \in \hat{\mathbb{G}}^{k\times 2}$;

and then updates proofs in the following way; for each pairing-product equation of index ℓ, of the form: $\langle \boldsymbol{A}; \boldsymbol{\mathfrak{X}} \rangle \langle \boldsymbol{X}; \boldsymbol{\mathfrak{A}} \rangle \boldsymbol{X}^T \underline{\Gamma} \boldsymbol{\mathfrak{X}} = e(G, \mathfrak{G}_\ell) \in \mathbb{G}_T$, defined by: $\boldsymbol{A}_\ell \in \mathbb{G}^k, \boldsymbol{\mathfrak{A}}_\ell \in \hat{\mathbb{G}}^n, \underline{\Gamma}_\ell \in \mathbb{Z}_p^{n\times k}$, and $T_{T,\ell} = e(G, \mathfrak{G}_\ell) \in \mathbb{G}_T$, it draws $\underline{\widetilde{T}}_\ell \xleftarrow{\$} \mathbb{Z}_p^{2\times 2}$, and sets:

$$\underline{\widetilde{\pi}}_\ell \leftarrow \underline{\pi}_\ell + \underline{\widetilde{R}}^T \cdot \left(\boldsymbol{\mathfrak{A}}_\ell \cdot (0,1) + \underline{\Gamma}_\ell \widetilde{\mathbf{Com}}_{\boldsymbol{\mathfrak{X}}} \right) - \underline{\widetilde{T}}_\ell^T \begin{pmatrix} \boldsymbol{\mathfrak{U}}_1^T \\ \boldsymbol{\mathfrak{U}}_2^T \end{pmatrix} \in \hat{\mathbb{G}}^{2\times 2}$$

$$\underline{\widetilde{\theta}}_\ell \leftarrow \underline{\theta}_\ell + \underline{\widetilde{\mathfrak{R}}}^T \cdot \left(\boldsymbol{A}_\ell \cdot (0,1) + \underline{\Gamma}_\ell^T \mathbf{Com}_{\boldsymbol{X}} \right) + \underline{\widetilde{T}}_\ell \begin{pmatrix} \boldsymbol{U}_1^T \\ \boldsymbol{U}_2^T \end{pmatrix} \in \mathbb{G}^{2\times 2};$$

Finally, $\widetilde{\boldsymbol{\Pi}}$ is set as $((\underline{\widetilde{\pi}}_1, \underline{\widetilde{\theta}}_1), \dots, (\underline{\widetilde{\pi}}_L, \underline{\widetilde{\theta}}_L))$, considering the equation indices ℓ ranged from 1 to L. In the particular case where the equations are linear and the optimization from [GS08] is used, the randomization follows straightforwardly.
The algorithm returns: $\widetilde{\pi} \leftarrow (\widetilde{\mathbf{Com}}_{\boldsymbol{X}} \| \widetilde{\mathbf{Com}}_{\boldsymbol{\mathfrak{X}}}, \widetilde{\boldsymbol{\Pi}}, \mathcal{E})$.

$\mathsf{Vf}(\mathsf{pp}, \mathsf{crs}, \pi) \rightarrow b$**:** parsing the proof π as $(\mathbf{Com}_{\boldsymbol{X}} \| \mathbf{Com}_{\boldsymbol{\mathfrak{X}}}, \boldsymbol{\Pi}, \mathcal{E})$, for each pairing-product equation in $\mathcal{E}$ of index ℓ, defined by $\boldsymbol{A}_\ell \in \mathbb{G}^k, \boldsymbol{\mathfrak{A}}_\ell \in \hat{\mathbb{G}}^n, \underline{\Gamma}_\ell \in \mathbb{Z}_p^{n\times k}, T_{T,\ell} \in \mathbb{G}_T$, the algorithm verifies that:

$$\begin{aligned}((\boldsymbol{A} \cdot (0,1)) \bullet \mathbf{Com}_{\boldsymbol{\mathfrak{X}}}) \odot (\mathbf{Com}_{\boldsymbol{X}} \bullet (\boldsymbol{\mathfrak{A}} \cdot (0,1))) \odot (\mathbf{Com}_{\boldsymbol{X}} \bullet \underline{\Gamma}_\ell \mathbf{Com}_{\boldsymbol{\mathfrak{X}}}) \\ = \begin{pmatrix} 1 & 1 \\ 1 & T_{T,\ell} \end{pmatrix} \odot \left(\begin{pmatrix} \boldsymbol{U}_1^T \\ \boldsymbol{U}_2^T \end{pmatrix} \bullet \underline{\pi}_\ell \right) \odot \left(\underline{\theta}_\ell \bullet \begin{pmatrix} \boldsymbol{\mathfrak{U}}_1^T \\ \boldsymbol{\mathfrak{U}}_2^T \end{pmatrix} \right),\end{aligned}$$

where $\underline{B} \bullet \underline{\mathfrak{B}}$ denotes, for $\underline{B} = (B_{i,j})_{i,j} \in \mathbb{G}^{m\times 2}, \underline{\mathfrak{B}} = (\mathfrak{B}_{i,j})_{i,j} \in \hat{\mathbb{G}}^{m\times 2}$:

$$\underline{B} \bullet \underline{\mathfrak{B}} \leftarrow \begin{pmatrix} \prod_{i=1}^m e(B_{i,1}, \mathfrak{B}_{i,1}) & \prod_{i=1}^m e(B_{i,1}, \mathfrak{B}_{i,2}) \\ \prod_{i=1}^m e(B_{i,2}, \mathfrak{B}_{i,1}) & \prod_{i=1}^m e(B_{i,2}, \mathfrak{B}_{i,2}) \end{pmatrix} \in \mathbb{G}_T^{2\times 2}.$$

If any of the equation checks does not pass, the algorithm returns $b \leftarrow 0$; else, it returns $b \leftarrow 1$.

B Deferred Proofs

B.1 Proof of the Verifiability of TREnc

Proof. This theorem is first proven for the fresh-ciphertext verifiability, then the randomized-ciphertext one, using in each case a sequence of games, given a security parameter $\lambda \in \mathbb{N}$; but first of all, we show that the key generation algorithm may be simulated in an indistinguishable way allowing to check whether ciphertexts are in the encryption or randomization ranges efficiently:

– a challenger $\mathcal{C}$ sets $(\mathsf{PK}_0, \mathsf{sk}_0) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$, and $(\mathsf{PK}_1, \mathsf{sk}_1)$ drawn also following the Gen algorithm for the input λ, except that now, when generating PK_1, the challenger uses the Groth-Sahai CRS: $\boldsymbol{\varphi}_1 \xleftarrow{\$} \mathsf{GS.BCRSGen(pp)}$ instead of PK_0's $\boldsymbol{\varphi}_0 \xleftarrow{\$} \mathsf{GS.HCRSGen(pp)}$, generating this common reference string in a now extractable mode, by keeping the trapdoor scalars a and $\mathfrak{a}$ providing the factor between $\boldsymbol{\varphi}_0$'s first and second components in memory; this is denoted with: $(\mathsf{PK}_1, \mathsf{sk}_1, (a, \mathfrak{a})) \xleftarrow{\$} \mathsf{SimGen}(1^\lambda)$;
– $\mathcal{C}$ then draws $b \xleftarrow{\$} \{0; 1\}$ and returns $(\mathsf{PK}_b, \mathsf{sk}_b)$ to a PPT adverary $\mathcal{A}$, who answers with a guess b', and winning if it is equal to b.
– the difference between the distributions of $(\mathsf{PK}_0, \mathsf{sk}_0)$ and $(\mathsf{PK}_1, \mathsf{sk}_1)$ is two SXDH distinguishers, so $\mathcal{A}$'s advantage is this game is bounded by $2 \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathsf{SXDH}}(\lambda)$.

Then, the sequence of game is described from this simulated key generation:

Game $\mathbf{G}_0$: is the original security game for the verifiability of fresh ciphertexts. A challenger $\mathcal{C}$ sets $(\mathsf{PK}, \mathsf{sk}, (a, \mathfrak{a})) \xleftarrow{\$} \mathsf{SimGen}(1^\lambda)$, then sends over $(\mathsf{PK}, \mathsf{sk})$ to a PPT adversary $\mathcal{A}$; $\mathcal{A}$ then replies with C, and wins the game if $\mathsf{Ver}(\mathsf{PK}, C, \mathsf{fresh}) = 1$ and $C \notin \mathcal{C}_0$; $\mathcal{C}$ can efficiently check whether an element is in $\mathcal{C}_0$ using sk, as the PQPKE decryption operation yields the unique decomposition of C's $(\mathsf{c}, \mathsf{c}')$ component with respect to p, p' and Δ as $\mathsf{u}, \mathsf{m}, \mathsf{e}, \mathsf{e}'$ in $(\mathsf{c}, \mathsf{c}') = (\mathsf{p} \cdot \mathsf{u} + \Delta[\mathsf{m}]_t + \mathsf{e}, \mathsf{p}' \cdot \mathsf{u} + \mathsf{e}')$ such that $\mathsf{e}, \mathsf{e}' \in \mathsf{pol}_{\mathcal{R}_p}(\llbracket \Delta \rrbracket^n)$ and $\mathsf{m} \in \mathsf{pol}_{\mathcal{R}_p}(\llbracket t \rrbracket^n)$, and $(a, \mathfrak{a})$ allows the extraction of elements committed using the common reference string of PK. $\mathcal{A}$'s winning probability in this game is denoted: $\Pr\left\{S_{\mathcal{A}}^{\mathbf{G}_0}(\lambda) = 1\right\} = \Pr\{\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Ver,fresh}}(\lambda) = 1\}$.

Game $\mathbf{G}_1$: is identical to the previous game, except that now, in the final step deciding on the adversary's success, if $\mathsf{Ver}(\mathsf{PK}, C, \mathsf{fresh}) = 1$, the challenger extracts the value committed to in the $\mathsf{Com}_{\mathfrak{B}}$ component of Com, $\mathfrak{b}\mathfrak{G}$, and declares the game lost by $\mathcal{A}$ if $\mathfrak{b} \neq 1$.
If $\mathfrak{b} \neq 1$, then $\mathcal{C}$ extracts the values committed to in $\mathsf{Com}_{\sigma,\tau}$ and Com_τ inside of Com, $\sigma_\tau = (\Sigma_1, \Sigma_2)$ and (T_1, T_2) respectively. The perfect soundness of proofs made with the perfectly binding $\boldsymbol{\varphi}$ then ensures that:

$$\begin{cases} e(S_1, \mathfrak{G})e(S_2, \mathfrak{H}) = e(T_1, \mathfrak{G}_{\mathsf{spk},1})e(T_2, \mathfrak{G}_{\mathsf{spk},2}) \\ e(T_1, \mathfrak{G}) = e((1 - \mathfrak{b})G, \mathfrak{G}) \\ e(T_2, \mathfrak{G}) = e(\tau(1 - \mathfrak{b})G, \mathfrak{G}) \end{cases}$$

$$e(S_1, \mathfrak{G})e(S_2, \mathfrak{H}) = e((1-\mathfrak{b})G, \mathfrak{G}_{\mathsf{spk},1})e(\tau(1-\mathfrak{b})G, \mathfrak{G}_{\mathsf{spk},2})$$

with $1 - \mathfrak{b} \neq 0$, which would yield an efficient SXDH solver. Thus:

$$\left|\Pr\{S_{\mathcal{A}}^{\mathbf{G}_1}(\lambda)\} - \Pr\{\mathsf{Exp}_{\mathcal{A}}^{\mathbf{G}_0}(\lambda)\}\right| \leqslant \mathsf{Adv}_{\mathcal{A}}^{\mathsf{SXDH}}(\lambda).$$

Game $\mathbf{G}_2$: is as the previous game, but now, if $\mathsf{Ver}(\mathsf{PK}, C, \mathsf{fresh}) = 1$, the challenger extracts, from Com's $\mathsf{Com}_{\sigma,1}$ component, and using a, the value σ, and then checks, whether the vector $\boldsymbol{C} \leftarrow (c_1, \dots, c_n, c_1', \dots, c_n', 1, 0, \dots, 0) \cdot G \in \mathbb{G}_p^{2n+1+\widetilde{N}}$ (obtained straightforwardly from C's $(\mathsf{c}, \mathsf{c}')$ component), is such that $\mathsf{LHSP.Ver}(\boldsymbol{C}, \sigma, \mathsf{opk})$ outputs 1 – if not, $\mathcal{C}$ declares the game lost by $\mathcal{A}$.
She also extracts σ_i from each other $\mathsf{Com}_{\sigma,i}$ commitment, and verifies that $\mathsf{LHSP.Ver}((\widetilde{\boldsymbol{p}}_{i-1} \| \boldsymbol{e}_{i+1})G, \sigma_i, \mathsf{opk}) = 1$, denoting $(\boldsymbol{e}_i)_i$ the canonical basis of $\mathbb{Z}_p^{(1+\widetilde{N})\times(1+\widetilde{N})}$.
She then proceeds to extract $\boldsymbol{W} = (W_i)_i$ and $\boldsymbol{\mathfrak{W}} = (\mathfrak{W}_i)_i$ from $\mathsf{Com}_{\boldsymbol{W}}$ and $\mathsf{Com}_{\boldsymbol{\mathfrak{W}}}$. For each index i, she verifies that: $e(W_i - G, \mathfrak{W}_i) = 1_T$, $e(W_i, \mathfrak{G}) = e(G, \mathfrak{W}_i)$ (which shows that there exists $w_i \in \{0;1\}$ such that $W_i = w_i G$ and $\mathfrak{W}_i = w_i \mathfrak{G}$). For each $j \in [\![1;n]\!]$, she checks that:

$$\prod_{i=1}^{N} e(W_i, \mathfrak{G})^{p_{i,j}} = e(c_j G, \mathfrak{G}) \qquad \prod_{i=1}^{N} e(W_i, \mathfrak{G})^{p_{i,j+n}} = e(c_j' G, \mathfrak{G})$$

If any of the above tests failed to pass, the challenger declares that the adversary lost the game. The perfectly binding property of $\boldsymbol{\varphi}$ ensures that they never fail after $\mathsf{Ver}(\mathsf{PK}, C, \mathsf{fresh})$ has passed with $\mathfrak{b} = 1$, so the adversary's success probability is unchanged: $\Pr\{S_{\mathcal{A}}^{\mathbf{G}_2}(\lambda)\} = \Pr\{S_{\mathcal{A}}^{\mathbf{G}_1}(\lambda)\}$.
In this last game, the challenger has found, as $\boldsymbol{W}$ was shown to have components in $\{O, G\}$, a vector of bits $\boldsymbol{w}$ such that $\boldsymbol{W} = \boldsymbol{w}G$, and equivalently $\mathsf{m} \in \mathsf{pol}_{\mathcal{R}_t}(\{0;1\}^n), \mathsf{u} \in \mathsf{pol}_{\mathcal{R}_p}(\{0;1\}^n)$, and $\mathsf{e}, \mathsf{e}_0' \in \mathsf{pol}_{\mathcal{R}_p}([\![B]\!])$ such that: $(\mathsf{c}, \mathsf{c}') = (\mathsf{p} \cdot \mathsf{u} + \Delta[\mathsf{m}]_t + \mathsf{e}, \mathsf{p}' \cdot \mathsf{u} + \mathsf{e}')$, which means that C is in $\mathcal{C}_0$, and thus the adversary is incapable of winning.

Finally: $\Pr\left\{\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Ver},\mathsf{fresh}}(\lambda) = 1\right\} \leqslant \mathsf{Adv}_{\mathcal{A}}^{\mathsf{SXDH}}(\lambda)$.

The second proof for the randomized ciphertext verifiability is very similar to the above one, except that more bits are committed and extracted to make room for the bigger $\mathcal{C}_1$ space, according to the sequence of games presented in Appendix B.1.

From the above sequences of games: $\Pr\left\{\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Ver},\mathsf{TREnc}}(\lambda) = 1\right\} \leqslant \mathsf{Adv}_{\mathcal{A}}^{\mathsf{SXDH}}(\lambda)$, so finally, the verifiability from the fresh and randomized cases, and the distinguishing advatage between the honest and simulated key generations is granted with: $\Pr\left\{\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Ver}}(\lambda) = 1\right\} \leqslant 4 \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathsf{SXDH}}(\lambda)$.

End of the Verifiability Proof of TREnc

Game $\mathbf{G}_0$: is the original security game for the verifiability of randomized ciphertexts. A challenger $\mathcal{C}$ sets $(\mathsf{PK}, \mathsf{sk}, (a, \mathfrak{a})) \xleftarrow{\$} \mathsf{SimGen}(1^\lambda)$, then sends over $(\mathsf{PK}, \mathsf{sk})$ to a PPT adversary $\mathcal{A}$; $\mathcal{A}$ then replies with C, and wins the game if $\mathsf{Ver}(\mathsf{PK}, C, \mathsf{rand}) = 1$ and $C \notin \mathcal{C}_1$; $\mathcal{C}$ can efficiently check whether an element is in $\mathcal{C}_1$ using sk, as the PQPKE decryption operation yields the unique decomposition of C's $(\mathsf{c}, \mathsf{c}')$ component with respect to p, p' and $\varDelta$ as $\mathsf{u}, \mathsf{m}, \mathsf{e}, \mathsf{e}'$ in $(\mathsf{c}, \mathsf{c}') = (\mathsf{p} \cdot \mathsf{u} + \varDelta[\mathsf{m}]_t + \mathsf{e}, \mathsf{p}' \cdot \mathsf{u} + \mathsf{e}')$ such that $\mathsf{e}, \mathsf{e}' \in \mathsf{pol}_{\mathcal{R}_p}(\llbracket \varDelta \rrbracket^n)$ and $\mathsf{m} \in \mathsf{pol}_{\mathcal{R}_p}(\llbracket t \rrbracket^n)$, and the trapdoor $(a, \mathfrak{a})$ allows the extraction of all values committed to in GS proofs using PK's CRS. $\mathcal{A}$'s winning probability in this game is denoted: $\Pr\left\{S_{\mathcal{A}}^{\mathbf{G}_0}(\lambda) = 1\right\} = \Pr\left\{\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Ver},\mathsf{rand}}(\lambda) = 1\right\}$.

Game $\mathbf{G}_1$: is identical to the previous game, except that now, in the final step deciding on the adversary's success, if $\mathsf{Ver}(\mathsf{PK}, C, \mathsf{rand}) = 1$, the challenger extracts the value committed to in the $\mathsf{Com}_{\mathfrak{B}}$ component of Com, $\mathfrak{b}\mathfrak{G}$, and declares the game lost by $\mathcal{A}$ if $\mathfrak{b} \neq 1$.

If $\mathfrak{b} \neq 1$, then $\mathcal{C}$ extracts the values committed to in $\mathsf{Com}_{\sigma,\tau}$ and Com_τ inside of Com, $\sigma_\tau = (\varSigma_1, \varSigma_2)$ and (T_1, T_2) respectively. The perfect soundness of proofs made with the perfectly binding $\boldsymbol{\varphi}$ then ensures that:

$$\begin{cases} e(S_1, \mathfrak{G}) e(S_2, \mathfrak{H}) = e(T_1, \mathfrak{G}_{\mathsf{spk},1}) e(T_2, \mathfrak{G}_{\mathsf{spk},2}) \\ e(T_1, \mathfrak{G}) = e((1 - \mathfrak{b})G, \mathfrak{G}) \\ e(T_2, \mathfrak{G}) = e(\tau(1 - \mathfrak{b})G, \mathfrak{G}) \end{cases}$$

$$e(S_1, \mathfrak{G}) e(S_2, \mathfrak{H}) = e((1 - \mathfrak{b})G, \mathfrak{G}_{\mathsf{spk},1}) e(\tau(1 - \mathfrak{b})G, \mathfrak{G}_{\mathsf{spk},2})$$

with $1 - \mathfrak{b} \neq 0$, which would yield an efficient SXDH distinguisher. Thus:

$$\left| \Pr\{S_{\mathcal{A}}^{\mathbf{G}_1}(\lambda)\} - \Pr\{\mathsf{Exp}_{\mathcal{A}}^{\mathbf{G}_0}(\lambda)\} \right| \leqslant \mathsf{Adv}_{\mathcal{A}}^{\mathsf{SXDH}}(\lambda).$$

Game $\mathbf{G}_2$: is as the previous game, but now, if $\mathsf{Ver}(\mathsf{PK}, C, \mathsf{rand}) = 1$, the challenger extracts, from Com's Com_σ component, and using a, the value σ, and from Com's $\mathsf{Com}_{\widetilde{\boldsymbol{W}}}$ component, the vector $\widetilde{\boldsymbol{W}} = (\widetilde{W}_1, \ldots, \widetilde{W}_{\widetilde{N}})$, and then checks, whether the vector $\boldsymbol{C} \leftarrow (c_1, \ldots, c_n, c_1', \ldots, c_n', 1) \cdot G \| \widetilde{\boldsymbol{W}} \in \mathbb{G}_p^{2n+1+\widetilde{N}}$ (obtained straightforwardly from C's $(\mathsf{c}, \mathsf{c}')$ component), is such that $\mathsf{LHSP.Ver}(\boldsymbol{C}, \sigma, \mathsf{opk})$ outputs 1 – if not, $\mathcal{C}$ declares the game lost by $\mathcal{A}$. She then proceeds to extract $\boldsymbol{W} = (W_i)_i$, $\boldsymbol{\mathfrak{W}} = (\mathfrak{W}_i)_i$ and $\widetilde{\boldsymbol{\mathfrak{W}}} = (\widetilde{\mathfrak{W}}_i)_i$ from $\mathsf{Com}_{\boldsymbol{W}\|\widetilde{\boldsymbol{W}}}$ and $\mathsf{Com}_{\boldsymbol{\mathfrak{W}}\|\widetilde{\boldsymbol{\mathfrak{W}}}}$.

For each index i, she verifies that: $e(W_i - G, \mathfrak{W}_i) = 1_T$, $e(W_i, \mathfrak{G}) = e(G, \mathfrak{W}_i)$ (which shows that $W_i \in \{O, G\}$), and similarly, for each index k that: $e(\widetilde{W}_i - G, \widetilde{\mathfrak{W}}_i) = e(G, \mathfrak{G})$ and $e(\widetilde{W}_i, \mathfrak{G}) = e(G, \widetilde{\mathfrak{W}}_i)$. For each $j \in \llbracket 1; n \rrbracket$, she checks that:

$$\prod_{i=1}^{N} e(W_i, \mathfrak{G})^{p_{i,j}} \cdot \prod_{i=1}^{\widetilde{N}} e(\widetilde{W}_i, \mathfrak{G})^{\widetilde{p}_{i,j}} = e(c_j G, \mathfrak{G})$$

$$\prod_{i=1}^{N} e(W_i, \mathfrak{G})^{p_{i,j+n}} \cdot \prod_{i=1}^{\widetilde{N}} e(\widetilde{W}_i, \mathfrak{G})^{\widetilde{p}_{i,j+n}} = e(c_j' G, \mathfrak{G})$$

If any of the above tests failed to pass, the challenger declares that the adversary lost the game. The perfectly binding property of $\boldsymbol{\varphi}$ ensures that they never fail after $\mathsf{Ver}(\mathsf{PK}, C, \mathsf{rand})$ has passed with $\mathfrak{b} = 1$, so the adversary's success probability is unchanged: $\Pr\{S_{\mathcal{A}}^{\mathbf{G}_2}(\lambda)\} = \Pr\{S_{\mathcal{A}}^{\mathbf{G}_1}(\lambda)\}$.
In this last game, the challenger has found, since $\boldsymbol{W}\|\widetilde{\boldsymbol{W}}$ is a vector whose components are in $\{O, G\}$, a vector of bits $\boldsymbol{w}\|\widetilde{\boldsymbol{w}}$ such that $\boldsymbol{W}\|\widetilde{\boldsymbol{W}} = (\boldsymbol{w}\|\widetilde{\boldsymbol{w}})G$, and equivalently, $\mathsf{m}_0 \in \mathsf{pol}_{\mathcal{R}_t}(\{0;1\}^n)$, $\mathsf{u}_0, \mathsf{u}_1 \in \mathsf{pol}_{\mathcal{R}_p}(\{0;1\}^n)$, and $\mathsf{e}_0, \mathsf{e}_1, \mathsf{e}_0', \mathsf{e}_1' \in \mathsf{pol}_{\mathcal{R}_p}([\![B]\!])$ such that:

$$(\mathsf{c}, \mathsf{c}') = (\mathsf{p} \cdot (\mathsf{u}_0 + \mathsf{u}_1) + \Delta[\mathsf{m}_0]_t + \mathsf{e}_0 + \mathsf{e}_1, \mathsf{p}' \cdot (\mathsf{u} + \mathsf{u}_1) + \mathsf{e}'_0 + \mathsf{e}'_1),$$

which means that C is in $\mathcal{C}_1$, and thus the adversary is incapable of winning.

Finally: $\Pr\left\{\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Ver},\mathsf{rand}}(\lambda) = 1\right\} \leqslant \mathsf{Adv}_{\mathcal{A}}^{\mathsf{SXDH}}(\lambda)$.

B.2 Proof of the Traceability of TREnc

Proof. The following sequence of game reduces the traceability of the scheme to a game in which the adversary's best strategy is to provide a random output.

Game $\mathbf{G}_0$: is the initial traceability security game for $\mathsf{TREnc} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Trace}, \mathsf{Rand}, \mathsf{Ver}, \mathsf{Dec})$. Let $\mathcal{A}$ be a stateful PPT adversary, $\lambda \in \mathbb{N}$, $(\mathsf{PK}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$. $\mathcal{A}_1$ is provided with $(\mathsf{PK}, \mathsf{sk})$, has access to an encryption oracle provided by $\mathcal{C}$, and queries Q messages in $\{0;1\}^n$. The challenger, $\mathcal{C}$, then sets, for each query of index i, $\boldsymbol{m}_i \in \{0;1\}^n$: $(\mathsf{osk}_i, C_i) \leftarrow \mathsf{Enc}(\mathsf{PK}, \boldsymbol{m}_i)$, and returns C_i to $\mathcal{A}$. At one point $\mathcal{A}$ returns the ciphertext C^*, and wins the game iff there exists an index $i \in [\![1;Q]\!]$ such that $\mathsf{Trace}(\mathsf{PK}, C_i) = \mathsf{Trace}(\mathsf{PK}, C^*)$, there exists $\ell \in \{\mathsf{fresh}, \mathsf{rand}\}$ such that $\mathsf{Ver}(\mathsf{PK}, C^*, \ell) = 1$, and $\mathsf{Dec}(\mathsf{sk}, C^*) \neq \boldsymbol{m}_i$, which happens with probability $\Pr\{S_{\mathcal{A}}^{\mathbf{G}_0}(\lambda)\} = \Pr\{\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Trace}}(\lambda)\}$.

Game $\mathbf{G}_1$: is exactly the same as the initial game, except that now, when generating PK, the challenger uses the Groth-Sahai CRS: $\boldsymbol{\varphi} \xleftarrow{\$} \mathsf{GS.BCRSGen}(\mathsf{pp})$ instead of the previous game's $\boldsymbol{\varphi} \xleftarrow{\$} \mathsf{GS.HCRSGen}(\mathsf{pp})$, generating this common reference string in a now extractable mode.
In this new game, the components of the vectors in $\boldsymbol{\varphi}$ form a Diffie-Hellman tuple in $\mathbb{G}$ and another one in $\hat{\mathbb{G}}$, whereas they did not in $\mathbf{G}_0$.
The difference between both games is two SXDH distinguishers, and the probability $\Pr\{S_{\mathcal{A}}^{\mathbf{G}_1}(\lambda)\}$ that $\mathcal{A}$ will win the current game is thus bounded with: $|\Pr\{S_{\mathcal{A}}^{\mathbf{G}_1}(\lambda)\} - \Pr\{\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Trace}}(\lambda)\}| \leqslant 2 \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathsf{SXDH}}(\lambda)$.

Game $\mathbf{G}_2$: is identical to the previous game, except that now, in the execution of $\mathsf{GS.BCRSGen}(\mathsf{pp})$ to generate $\boldsymbol{\varphi}$, after a and $\mathfrak{a}$ are drawn at random in $\mathbb{Z}_p$ to set second vector components as this multiple of the first ones (a for elements in $\mathbb{G}$ and $\mathfrak{a}$ for elements in $\hat{\mathbb{G}}$), a and $\mathfrak{a}$ are kept by the challenger as a trapdoor that will allow to extract elements committed using this common reference string. The distribution of what is send to the adversary is not changed, but now in the final step deciding on the adversary's success, if $\mathsf{opk}_q = \mathsf{opk}^*$, $\mathsf{Dec}(\mathsf{sk}, C^*) \neq \boldsymbol{m}_q$, and there is an $\ell \in \{\mathsf{fresh}, \mathsf{rand}\}$ such that

$\mathsf{Ver}(\mathsf{PK}, C^*, \ell) = 1$, when proceeding to the verification of C^*, the challenger extracts the value committed to in the $\mathsf{Com}^*_{\mathfrak{B}}$ component of Com, $\mathfrak{b}^*\mathfrak{G}$, and declares the game lost by $\mathcal{A}$ if $\mathfrak{b}^* \neq 1$.
If $\mathfrak{b}^* \neq 1$, then $\mathcal{C}$ extracts the values committed to in $\mathsf{Com}^*_{\sigma,\tau}$ and Com^*_{τ} inside of Com, $\sigma^*_\tau = (\Sigma^*_1, \Sigma^*_2)$ and (T^*_1, T^*_2) respectively. The perfect soundness of proofs made with the perfectly binding $\boldsymbol{\varphi}$ then ensures that:

$$\begin{cases} e(S^*_1, \mathfrak{G})e(S^*_2, \mathfrak{H}) = e(T^*_1, \mathfrak{G}_{\mathsf{spk},1})e(T^*_2, \mathfrak{G}_{\mathsf{spk},2}) \\ e(T^*_1, \mathfrak{G}) = e((1 - \mathfrak{b}^*)G, \mathfrak{G}) \\ e(T^*_2, \mathfrak{G}) = e(\tau(1 - \mathfrak{b}^*)G, \mathfrak{G}) \end{cases}$$
$$e(S^*_1, \mathfrak{G})e(S^*_2, \mathfrak{H}) = e((1 - \mathfrak{b}^*)G, \mathfrak{G}_{\mathsf{spk},1})e(\tau(1 - \mathfrak{b}^*)G, \mathfrak{G}_{\mathsf{spk},2})$$

with $1 - \mathfrak{b}^* \neq 0$, finding such a signature happens with probability $\frac{1}{p}$ plus $\mathcal{A}$'s SXDH advantage. Thus: $|\Pr\{S^{\mathbf{G}_2}_{\mathcal{A}}(\lambda)\} - \Pr\{\mathsf{Exp}^{\mathbf{G}_1}_{\mathcal{A}}(\lambda)\}| \leqslant \frac{1}{p} + \mathsf{Adv}^{\mathsf{SXDH}}_{\mathcal{A}}(\lambda)$.

Game $\mathbf{G}_3$: is as the previous game, but now, if $\mathsf{Ver}(\mathsf{PK}, C^*, \mathsf{rand}) = 1$, $\mathcal{C}$ extracts $\widetilde{\boldsymbol{W}}^*$ from $\mathsf{Com}^*_{\widetilde{\boldsymbol{W}}}$; then, from the commitment Com^*'s Com^*_σ component if $\mathsf{Ver}(\mathsf{PK}, C^*, \mathsf{rand}) = 1$, or its $\mathsf{Com}^*_{\sigma,1}$ component if $\mathsf{Ver}(\mathsf{PK}, C^*, \mathsf{fresh}) = 1$, the challenger extracts, using a, σ^*, and then checks, if $\mathsf{Ver}(\mathsf{PK}, C^*, \mathsf{fresh}) = 1$ whether the vector $\boldsymbol{C}^* \leftarrow (c^*_1, \ldots, c^*_n, c'^*_1, \ldots, c'^*_n, 1, 0, \ldots, 0) \cdot G \in \mathbb{G}_p^{2n+1+\widetilde{N}}$ (obtained straightforwardly from C^*'s $(\mathsf{c}^*, \mathsf{c}'^*)$ component), and if $\mathsf{Ver}(\mathsf{PK}, C^*, \mathsf{rand}) = 1$, the vector $\boldsymbol{C}^* \leftarrow (c^*_1, \ldots, c^*_n, c'^*_1, \ldots, c'^*_n, 1) \cdot G \| \widetilde{\boldsymbol{W}}^* \in \mathbb{G}_p^{2n+1+\widetilde{N}}$, is such that $\mathsf{LHSP.Ver}(\boldsymbol{C}^*, \sigma^*, \mathsf{opk}^*)$ outputs 1 – if not, $\mathcal{C}$ declares the game lost by $\mathcal{A}$.
If $\mathsf{Ver}(\mathsf{PK}, C^*, \mathsf{fresh}) = 1$, she also extracts σ^*_i from each other $\mathsf{Com}^*_{\sigma,i}$ commitment, and verifies that $\mathsf{LHSP.Ver}((\widetilde{\boldsymbol{p}}_{i-1} \| \boldsymbol{e}_{i+1})G, \sigma^*_i, \mathsf{opk}^*) = 1$, denoting $(\boldsymbol{e}_i)_i$ the canonical basis of $\mathbb{Z}_p^{(1+\widetilde{N}) \times (1+\widetilde{N})}$.
She then proceeds to extract, if $\mathsf{Ver}(\mathsf{PK}, C^*, \mathsf{fresh}) = 1$, $\boldsymbol{W}^* = (W^*_i)_i$ and $\boldsymbol{\mathfrak{W}}^* = (\mathfrak{W}^*_i)_i$ from $\mathsf{Com}^*_{\boldsymbol{W}}$ and $\mathsf{Com}^*_{\boldsymbol{\mathfrak{W}}}$, and if $\mathsf{Ver}(\mathsf{PK}, C^*, \mathsf{rand}) = 1$, $\boldsymbol{W}^* = (W^*_i)_i, \widetilde{\boldsymbol{W}}^* = (\widetilde{W}^*_k)_k$, $\boldsymbol{\mathfrak{W}}^* = (\mathfrak{W}^*_i)_i$ and $\widetilde{\boldsymbol{\mathfrak{W}}}^* = (\widetilde{\mathfrak{W}}^*_i)_i$ from $\mathsf{Com}^*_{\boldsymbol{W}\|\widetilde{\boldsymbol{W}}}$ and $\mathsf{Com}^*_{\boldsymbol{\mathfrak{W}}\|\widetilde{\boldsymbol{\mathfrak{W}}}}$. For each index i, she verifies that: $e(W^*_i - G, \mathfrak{W}^*_i) = 1_T$, $e(W^*_i, \mathfrak{G}) = e(G, \mathfrak{W}^*_i)$ (which shows that $W^*_i \in \{O, G\}$), and additionally if $\mathsf{Ver}(\mathsf{PK}, C^*, \mathsf{rand}) = 1$, for each index k that: $e(\widetilde{W}^*_i - G, \widetilde{\mathfrak{W}}^*_i) = 1_T$ and $e(\widetilde{W}^*_i, \mathfrak{G}) = e(G, \widetilde{\mathfrak{W}}^*_i)$.
Then,

– if $\mathsf{Ver}(\mathsf{PK}, C^*, \mathsf{fresh}) = 1$, for each $j \in [\![1; n]\!]$, she checks that:

$$\prod_{i=1}^{N} e(W^*_i, \mathfrak{G})^{p_{i,j}} = e(c_j G, \mathfrak{G}) \qquad \prod_{i=1}^{N} e(W^*_i, \mathfrak{G})^{p_{i,j+n}} = e(c'_j G, \mathfrak{G});$$

– if $\mathsf{Ver}(\mathsf{PK}, C^*, \mathsf{rand}) = 1$, for each $j \in [\![1; n]\!]$, she checks that:

$$\prod_{i=1}^{N} e(W^*_i, \mathfrak{G})^{p_{i,j}} \cdot \prod_{i=1}^{\widetilde{N}} e(\widetilde{W}^*_i, \mathfrak{G})^{\widetilde{p}_{i,j}} = e(c_j G, \mathfrak{G})$$

$$\prod_{i=1}^{N} e(W_i^*, \mathfrak{G})^{p_{i,j+n}} \cdot \prod_{i=1}^{\widetilde{N}} e(\widetilde{W}_i^*, \mathfrak{G})^{\tilde{p}_{i,j+n}} = e(c'_j G, \mathfrak{G})$$

If any of the above tests failed to pass, the challenger declares that the adversary lost the game. The perfectly binding property of φ ensures that they never fail after $\mathsf{Ver}(\mathsf{PK}, C^*, \ell)$ has passed with $\mathfrak{b}^* = 1$, so the adversary's success probability is unchanged: $\Pr\{S_{\mathcal{A}}^{\mathbf{G}_3}(\lambda)\} = \Pr\{S_{\mathcal{A}}^{\mathbf{G}_2}(\lambda)\}$.

Game $\mathbf{G}_4$: is as the previous game, except that $\mathcal{C}$ now declares the game lost for $\mathcal{A}$ if, for any encryption requests $q, q' \in [\![1; Q]\!]$, $\mathsf{opk}_q = \mathsf{opk}_{q'}$. For any pair of indices, this event happens with probability $\frac{1}{p^{2n+1+\widetilde{N}}}$, indepentently of other pairs: $\left|\Pr\{S_{\mathcal{A}}^{\mathbf{G}_4}(\lambda)\} - \Pr\{S_{\mathcal{A}}^{\mathbf{G}_3}(\lambda)\}\right| \leqslant \frac{Q^2}{p^{2n+1+\widetilde{N}}} \leqslant 2^{-n-\widetilde{N}/2} \leqslant 2^{-2n}$ as $Q \leqslant 2^{\lambda}$ with $\mathcal{A}$ being polynomial time.

Game $\mathbf{G}_5$: is as the previous game, but now, denoting $q \in [\![1; Q]\!]$ the index such that $\mathsf{Trace}(C_q) = \mathsf{Trace}(C^*)$, $\boldsymbol{w}^* = (w_i^*)_i$ the vector of bits such that $\boldsymbol{W}^* = \boldsymbol{w}^* G$, and if if $\mathsf{Ver}(\mathsf{PK}, C^*, \mathsf{rand}) = 1$, $\tilde{\boldsymbol{w}}^* = (\tilde{w}_k^*)_k$ the vector of bits such that $\widetilde{\boldsymbol{W}}^* = \tilde{\boldsymbol{w}}^* G$, $\mathcal{C}$ declares the game lost for $\mathcal{A}$ if:
- if $\mathsf{Ver}(\mathsf{PK}, C^*, \mathsf{fresh}) = 1$, $(c_1^*, \ldots, c_n'^*) \neq (c_1, \ldots, c'_n)$ (where $(c_1, \ldots, c'_n) = \mathsf{pol}_{\mathcal{R}_p}^{-1}(\mathsf{c}_q) \| \mathsf{pol}_{\mathcal{R}_p}^{-1}(\mathsf{c}'_q)$);
- if $\mathsf{Ver}(\mathsf{PK}, C^*, \mathsf{rand}) = 1$, $\sum_{i \in [\![1;N]\!]} w_i^* \cdot \boldsymbol{p}_i \neq (c_1, \ldots, c'_n)$, that is: if $\boldsymbol{C}^* \neq (c_1, \ldots, c'_n, 1, 0, \ldots, 0) + \sum_{k=1}^{\widetilde{N}} \tilde{w}_k^* (\tilde{\boldsymbol{p}}_k \| \boldsymbol{e}_{k+1}) = \boldsymbol{T}_{q,1} + \sum_{k=1}^{\widetilde{N}} \widetilde{w_k}^* \boldsymbol{T}_{k+1}$ (where $\boldsymbol{T}_{q,1}$ is the first vector of C_q's tracing matrix).

The above equalities pass if and only if $\boldsymbol{C}^*$ is in $\mathsf{span}\{\boldsymbol{T}_{q,1}, \ldots, \boldsymbol{T}_{q,\widetilde{N}+1}\}$, as the rightmost block of $\underline{T}$ is $\underline{\mathsf{Id}}_{\widetilde{N}+1}$, the identity matrix of $\mathbb{Z}_p^{(\widetilde{N}+1)\times(\widetilde{N}+1)}$, and $\tilde{\boldsymbol{w}} G = \widetilde{\boldsymbol{W}}$.

If $\mathcal{C}^*$ were not in $\mathsf{span}\{\boldsymbol{T}_{q,1}, \ldots, \boldsymbol{T}_{q,\widetilde{N}+1}\}$, then when $\mathsf{LHSP.Ver}(\boldsymbol{C}^*, \sigma^*, \mathsf{opk}^*) = 1$, $\boldsymbol{C}^*$ and σ^* would yield an SXDH solver, so: $\left|\Pr\{S_{\mathcal{A}}^{\mathbf{G}_5}(\lambda)\} - \Pr\{\mathsf{Exp}_{\mathcal{A}}^{\mathbf{G}_4}(\lambda)\}\right| \leqslant \mathsf{Adv}_{\mathcal{A}}^{\mathsf{SXDH}}(\lambda)$.

Game $\mathbf{G}_6$: is as the previous game, except that now, $\mathcal{C}$ declares the game lost for $\mathcal{A}$ if $\mathsf{Dec}(\mathsf{sk}, C^*) \neq \boldsymbol{m}_q$. As $\mathcal{C}$ has obtained $(w_i^*)_i, (\widetilde{w_k}^*)_k$ such that $(c_1^*, \ldots, c_n'^*) = \sum_{i \in [\![1;N]\!]} w_i^* \boldsymbol{p}_i + \sum_{k \in [\![1;\widetilde{N}]\!]} \widetilde{w_k}^* \tilde{\boldsymbol{p}}_k$, $(c_1, \ldots, c'_n) = \sum_{i \in [\![1;N]\!]} w_i^* \boldsymbol{p}_i$, and verified that the w_i^*'s and $\widetilde{w_i}^*$'s are in $\{0; 1\}$, then it equivalently means that she knows $\mathsf{m}^* \in \mathsf{pol}_{\mathcal{R}_t}(\{0;1\}^n)$, $\mathsf{u}_0^*, \mathsf{u}_1^* \in \mathsf{pol}_{\mathcal{R}_p}(\{0;1\}^n)$, $\mathsf{e}_0^*, \mathsf{e}_0'^* \in \mathsf{pol}_{\mathcal{R}_p}$ $([\![2B]\!])$, and $\mathsf{e}_1^*, \mathsf{e}_1'^* \in \mathsf{pol}_{\mathcal{R}_p}([\![2^{\lambda+1}B]\!])$ such that:

$$\begin{cases} (\mathsf{c}^*, \mathsf{c}'^*) = (\mathsf{p} \cdot (\mathsf{u}_0^* + \mathsf{u}_1^*) + \Delta[\mathsf{m}^*]_t + \mathsf{e}_0^* + \mathsf{e}_1^*, \mathsf{p}' \cdot (\mathsf{u}_0^* + \mathsf{u}_1^*) + \mathsf{e}_0'^* + \mathsf{e}_1'^*), \\ (\mathsf{c}_q, \mathsf{c}'_q) = (\mathsf{p} \cdot \mathsf{u}_0^* + \Delta[\mathsf{m}^*]_t + \mathsf{e}_0^*, \mathsf{p}' \cdot \mathsf{u}_0^* + \mathsf{e}_0'^*), \end{cases}$$

and thus, as $2B \leqslant \Delta$: $\mathsf{PQPKE.Dec}((\mathsf{c}^*, \mathsf{c}'^*)) = \mathsf{PQPKE.Dec}((\mathsf{c}_q, \mathsf{c}'_q)) = \mathsf{m}^*$, so $\mathsf{Dec}(\mathsf{sk}, C^*) = \mathsf{Dec}(\mathsf{sk}, C_q) = \boldsymbol{m}_q$ (under the correctness of the scheme), so: $\Pr\{S_{\mathcal{A}}^{\mathbf{G}_6}(\lambda)\} = \Pr\{\mathsf{Exp}_{\mathcal{A}}^{\mathbf{G}_5}(\lambda)\}$. Moreover, this ensures that $\mathcal{A}$ can never win the game, as $\mathsf{Dec}(\mathsf{sk}, C^*) = \boldsymbol{m}_q$, so in this last game: $\Pr\{S_{\mathcal{A}}^{\mathbf{G}_6}(\lambda)\} = 0$.

Finally: $\Pr\{\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Trace}}(\lambda)\} \leqslant 2^{-2n} + \frac{1}{p} + 4 \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathsf{SXDH}}(\lambda)$.

B.3 Proof of the TCCA-Security of TREnc

Proof. The following sequence of games reduces the TCCA one to one in which the challenge sent to the adversary does not depend on the target bit b she should get, thus leading to a null advantage.

Game $\mathbf{G}_0$: is the original TCCA-security game. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a PPT adversary, $\lambda \in \mathbb{N}$ a security parameter. The challenger, $\mathcal{C}$, runs $(\mathsf{PK}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$, then sends PK over to $\mathcal{A}_1$, who, having an oracle-access to the Dec algorithm, outputs a state st and two ciphertexts, C_0^* and C_1^*.
The challenger then checks that $\mathsf{Trace}(\mathsf{PK}, C_0^*) = \mathsf{Trace}(\mathsf{PK}, C_1^*)$, that $\mathsf{Vf}(\mathsf{PK}, C_0^*, \mathsf{fresh}) = 1$, and that $\mathsf{Vf}(\mathsf{PK}, C_1^*, \mathsf{fresh}) = 1$, declaring the game lost for $\mathcal{A}$ if any of the above conditions is not met; else, she goes on drawing $b \xleftarrow{\$} \{0; 1\}$, setting $C^* \xleftarrow{\$} \mathsf{Rand}(\mathsf{PK}, C_b^*)$, and returns C^* to $\mathcal{A}_2$, who, taking it as input along with the state st and having access to a decryption oracle returning results only when queried on ciphertexts C such that $\mathsf{Trace}(\mathsf{PK}, C) \neq \mathsf{Trace}(\mathsf{PK}, C^*)$, outputs a guess b' for the value of b; $\mathcal{A}$ then wins only if b' is equal to b, with a success probability denoted: $\Pr\{S_{\mathcal{A}}^{\mathbf{G}_0}(\lambda)\} = \Pr\{\mathsf{Exp}_{\mathcal{A}}^{\mathrm{TCCA}}(\lambda)\}$.

Game $\mathbf{G}_1$: is as the previous game, except that now, the challenger aborts and declares the game lost by the adversary if at any point $\mathcal{A}$ produces two ciphertexts (including as oracle queries) C and C' such that, denoting $\mathsf{opk} \leftarrow \mathsf{Trace}(\mathsf{PK}, C)$ and $\mathsf{opk}' \leftarrow \mathsf{Trace}(\mathsf{PK}, C')$, $\mathsf{opk} \neq \mathsf{opk}'$ but $\mathcal{H}(\mathsf{opk}, \mathsf{PK}) = \mathcal{H}(\mathsf{opk}', \mathsf{PK})$. This event happens only if a collision is found on $\mathcal{H}$, thus the adversary's success probability loss with respect to the previous game is bounded in the following way: $\left|\Pr\{S_{\mathcal{A}}^{\mathbf{G}_1}(\lambda)\} - \Pr\{S_{\mathcal{A}}^{\mathbf{G}_0}(\lambda)\}\right| \leqslant \mathsf{Adv}_{\mathcal{A}}^{\mathsf{coll}, \mathcal{H}}(\lambda)$.

Game $\mathbf{G}_2$: is as the previous game, except that now, in the execution of the Gen algorithm outputting $(\mathsf{PK}, \mathsf{sk})$, $\mathcal{C}$ now replaces the generation of spk as a random draw from $\hat{\mathbb{G}}^2$ by: $(\mathsf{spk}, \mathsf{ssk}) \xleftarrow{\$} \mathsf{LHSP.KeyGen}((\mathsf{pp}, \mathfrak{H}), 2)$, generating a signature key pair for which spk will be included in the common reference string in PK, while $\mathcal{C}$ keeps ssk in memory to be be able to sign elements with a corresponding key later on. The view of $\mathcal{A}$ is not altered since the $\mathsf{LHSP.KeyGen}$-generated spk also follows a uniform distribution on $\hat{\mathbb{G}}^2$, so $\Pr\{S_{\mathcal{A}}^{\mathbf{G}_2}(\lambda)\} = \Pr\{S_{\mathcal{A}}^{\mathbf{G}_1}(\lambda)\}$.

Game $\mathbf{G}_3$: is as the previous game, but now, when generating C^*, the challenger sets $\mathfrak{b} \leftarrow 0$, and uses the witness $\mathfrak{b}\mathfrak{G} \leftarrow \mathfrak{O}$ for the $\mathfrak{X}$ variable of the $\widetilde{\mathcal{E}}_{\mathsf{SiSo}}$ system of equations, along with $Y_i \leftarrow O$ for each index $i \in \left[\!\left[1; 2n + 1 + \widetilde{N}\right]\!\right]$, $(X_1, X_2) \leftarrow (O, O)$, for each $i \in [\![1; N]\!]$, $(Z_i, \mathfrak{Z}_i) \leftarrow (O, \mathfrak{O})$, and for each $k \in \left[\!\left[1; \widetilde{N}\right]\!\right]$, $(\widetilde{\mathfrak{Z}}_k, \widetilde{Z}_k) \leftarrow (\mathfrak{O}, O)$. $\mathcal{C}$ also sets: $(Y_1', Y_2') \leftarrow (G, \tau^* G)$ (where $\tau^* = \mathcal{H}(\mathsf{opk}^*, \mathsf{PK})$)), and generates a signature on this vector with: $\sigma_\tau^* \leftarrow \mathsf{LHSP.Sign}(\mathsf{ssk}, (G, \tau^* G))$, then finally setting $(X_1', X_2') \leftarrow \sigma_\tau^*$ as the final component of the solution w^* to $\widetilde{\mathcal{E}}_{\mathsf{SiSo}}$ that will be used; $\mathcal{C}$ then sets: $\mathit{\Pi}^* \leftarrow (\mathsf{Com}^*, \pi^*, \widetilde{\mathcal{E}}_{\mathsf{SiSo}}) \xleftarrow{\$} \mathsf{GS.Com\&Pr}(\mathsf{pp}, \varphi, w^*, \widetilde{\mathcal{E}}_{\mathsf{SiSo}})$, and, still using the same $(\mathsf{c}^*, \mathsf{c}'^*)$ and opk^* components of C^* as in the previous game, now sets: $C^* \leftarrow ((\mathsf{c}^*, \mathsf{c}'^*), \mathsf{opk}^*, \mathit{\Pi}^*)$ before returning C^* to $\mathcal{A}$.

The honest randomization of C_b^* was not modified, and as φ was generated in the Groth-Sahai perfectly witness-indistinguishable mode, the use of the new witness w^* in the Groth-Sahai proof leaves this game perfectly indistinguishable from the previous one: $\Pr\{S_{\mathcal{A}}^{\mathbf{G}_3}(\lambda)\} = \Pr\{S_{\mathcal{A}}^{\mathbf{G}_2}(\lambda)\}$.

Game $\mathbf{G}_4$: is as the previous game, except that now, the challenger uses: $\varphi \xleftarrow{\$} \mathsf{GS.BCRSGen}(\mathsf{pp})$ in the generation of the Groth-Sahai reference string included in PK, now using the perfectly binding mode.
In this new game, the components of the vectors in φ form a Diffie-Hellman tuple in $\mathbb{G}$ and another one in $\hat{\mathbb{G}}$, whereas they did not in $\mathbf{G}_3$.
The difference between both games is two SXDH distinguishers, and the probability $\Pr\{S_{\mathcal{A}}^{\mathbf{G}_4}(\lambda)\}$ that $\mathcal{A}$ will win the current game is thus bounded by: $\left|\Pr\{S_{\mathcal{A}}^{\mathbf{G}_4}(\lambda)\} - \Pr\{\mathsf{Exp}_{\mathcal{A}}^{\mathbf{G}_3}(\lambda)\}\right| \leqslant 2 \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathsf{SXDH}}(\lambda)$.

Game $\mathbf{G}_5$: is identical to the previous game except that now, when computing $\varphi \xleftarrow{\$} \mathsf{GS.BCRSGen}(\mathsf{pp})$, the challenger keeps in memory the trapdoor scalars a and $\mathfrak{a}$ that are the secret multiplicative factors of second vector components in $\mathbb{G}$ and $\hat{\mathbb{G}}$ respectively. The knowledge of a and $\mathfrak{a}$ will allow $\mathcal{C}$ to extract the unique values encoded in each Groth-Sahai commitment that is valid with respect to φ. The distribution of what is sent to $\mathcal{A}$ remains unchanged, so: $\Pr\{S_{\mathcal{A}}^{\mathbf{G}_5}(\lambda)\} = \Pr\{S_{\mathcal{A}}^{\mathbf{G}_4}(\lambda)\}$.

Game $\mathbf{G}_6$: is as the previous game, except that now, the challenger uses a and $\mathfrak{a}$ to extract the value encoded in the $\mathsf{Com}_{\mathfrak{B},k}$ component of Com_k, for each oracle query of index k, asking for the decryption of C_k such that there exists $\ell_k \in \{\mathsf{fresh}, \mathsf{rand}\}$ such that $\mathsf{Ver}(\mathsf{PK}, C_k, \ell_k) = 1$ and that $\mathsf{Trace}(\mathsf{PK}, C_k) \neq \mathsf{Trace}(\mathsf{PK}, C^*)$ if C^* has already been released (else the oracle answers with $\perp$). Indeed, as C_k passes the verification test, its commitments and proofs are valid, and the corresponding value $\mathfrak{b}_k\mathfrak{G}$ may be extracted using $\mathfrak{a}$. $\mathcal{C}$ now answers $\perp$ to the query if $\mathfrak{b}_k \neq 1$.
If $\mathfrak{b}_k \neq 1$, then $\mathcal{C}$ extracts the values committed to in $\mathsf{Com}_{\sigma,\tau,k}$ and $\mathsf{Com}_{\tau,k}$ inside of Com_k, denoting them $\sigma_{\tau,k} = (\Sigma_{k,1}, \Sigma_{k,2})$ and $(T_{k,1}, T_{k,2})$ respectively. The perfect soundness of proofs made with the perfectly binding φ then ensures that:

$$\begin{cases} e(S_{k,1}, \mathfrak{G})e(S_{k,2}, \mathfrak{H}) = e(T_{k,1}, \mathfrak{G}_{\mathsf{spk},1})e(T_{k,2}, \mathfrak{G}_{\mathsf{spk},2}) \\ e(T_{k,1}, \mathfrak{G}) = e((1-\mathfrak{b}_k)G, \mathfrak{G}) \\ e(T_{k,2}, \mathfrak{G}) = e(\tau_k(1-\mathfrak{b}_k)G, \mathfrak{G}) \end{cases}$$
$$e(S_{k,1}, \mathfrak{G})e(S_{k,2}, \mathfrak{H}) = e((1-\mathfrak{b}_k)G, \mathfrak{G}_{\mathsf{spk},1})e(\tau_k(1-\mathfrak{b}_k)G, \mathfrak{G}_{\mathsf{spk},2})$$

with $1 - \mathfrak{b}_k \neq 0$. If the corresponding index k was queried:

– after $\mathcal{A}$ sent C_0^* and C_1^* to $\mathcal{C}$, then it is valid only if $\mathsf{opk}_k \neq \mathsf{opk}^*$ and thus $\tau_k \neq \tau^*$; hence, if $\mathcal{A}$ were able to produce such a ciphertext with a probability not negligibly close to $\frac{1}{p}$, without having recieved any proof corresponding to τ_k (as $(G, \tau_k G)$ is then not in the span of $(G, \tau^* G)$), it would yield an efficient SXDH distinguisher;
– if it was queried before $\mathcal{A}$ sent C_0^* and C_1^* to $\mathcal{C}$, then the above argument also holds when $\tau_k \neq \tau^*$. If one had $\tau_k = \tau^*$, it also implies that $\mathcal{A}$ was

able to find an answer to the SXDH instance defined by spk; it is equal to the answer provided by ssk with probability $\frac{1}{p}$, and else, for a different basis decomposition, yields an efficient SXDH solver.

Thus:
$$\left|\Pr\{S_{\mathcal{A}}^{\mathbf{G}_6}(\lambda)\} - \Pr\{\mathsf{Exp}_{\mathcal{A}}^{\mathbf{G}_5}(\lambda)\}\right| \leqslant \frac{1}{p} + \mathsf{Adv}_{\mathcal{A}}^{\mathsf{SXDH}}(\lambda).$$

Game $\mathbf{G}_7$: is as the previous game, though now the challenger also extracts the following values:

- $\boldsymbol{W}_k = (W_{k,i})_i, \boldsymbol{\mathfrak{W}}_k = (\mathfrak{W}_{k,i})_i$ from $\mathsf{Com}_{k,\boldsymbol{W}}, \mathsf{Com}_{k,\boldsymbol{\mathfrak{W}}}$ respectively if $\mathsf{Ver}(\mathsf{PK}, C_k, \mathsf{fresh}) = 1$;
- $\boldsymbol{W}_k, \widetilde{\boldsymbol{W}}_k = (\widetilde{W}_{k,i})_i, \boldsymbol{\mathfrak{W}}_k, \widetilde{\boldsymbol{\mathfrak{W}}}_k = (\widetilde{\mathfrak{W}}_{k,i})_i$ from $\mathsf{Com}_{k,\boldsymbol{W}\|\widetilde{\boldsymbol{W}}}, \mathsf{Com}_{k,\boldsymbol{\mathfrak{W}}\|\widetilde{\boldsymbol{\mathfrak{W}}}}$ respectively if $\mathsf{Ver}(\mathsf{PK}, C_k, \mathsf{rand}) = 1$.

The prefect soundness of Groth-Sahai proofs verified with $\boldsymbol{\varphi}$ then ensures that, denoting $\boldsymbol{c}_k = (c_{k,1}, \ldots, c'_{k,n}) \leftarrow \mathsf{pol}_{\mathcal{R}_p}^{-1}(\mathsf{c}_k)\|\mathsf{pol}_{\mathcal{R}_p}^{-1}(\mathsf{c}'_k)$:

- if $\mathsf{Ver}(\mathsf{PK}, C_k, \mathsf{fresh}) = 1$:
$$\forall i \in [\![1; N]\!] : \qquad e(W_{k,i} - G, \mathfrak{W}_{k,i}) = 1_T \quad e(W_{k,i}, \mathfrak{G}) = e(G, \mathfrak{W}_{k,i})$$
so, as $\boldsymbol{W}_k$'s components are in $\{O, G\}$, this provides a vector of bits $(w_{k,1}, \ldots, w_{k,N})$ such that, because of the verified equations: $\sum_{i=1}^{N} w_{k,i} \cdot \boldsymbol{p}_i = \boldsymbol{c}_k$ and thus also yields $\mathsf{m}_k \in \mathsf{pol}_{\mathcal{R}_t}(\{0;1\}^n), \mathsf{u}_k \in \mathsf{pol}_{\mathcal{R}_p}(\{0;1\}^n)$, and $\mathsf{e}_k, \mathsf{e}'_k \in \mathsf{pol}_{\mathcal{R}_p}([\![2B]\!])$ such that:
$$(\mathsf{c}_k, \mathsf{c}'_k) = (\mathsf{p} \cdot \mathsf{u}_k + \Delta[\mathsf{m}_k]_t + \mathsf{e}_k, \mathsf{p}' \cdot \mathsf{u}_k + \mathsf{e}'_k);$$
- similarly if $\mathsf{Ver}(\mathsf{PK}, C_k, \mathsf{rand}) = 1$:
$$\forall i \in [\![1; N]\!] : \qquad e(W_{k,i} - G, \mathfrak{W}_{k,i}) = 1_T, \quad e(W_{k,i}, \mathfrak{G}) = e(G, \mathfrak{W}_{k,i})$$
$$\forall l \in \left[\!\!\left[1; \widetilde{N}\right]\!\!\right] : \qquad e(\widetilde{W}_{k,i} - G, \widetilde{\mathfrak{W}}_{k,i}) = 1_T, \quad e(\widetilde{W}_{k,i}, \mathfrak{G}) = e(G, \widetilde{\mathfrak{W}}_{k,i})$$
so $\boldsymbol{W}_k \in \{O, G\}^N$ and $\widetilde{\boldsymbol{W}}_k \in \{O, G\}^{\widetilde{N}}$, and they provide vectors of bits $\boldsymbol{w}_k = (w_{k,i})_i$ and $\widetilde{\boldsymbol{w}}_k = (\widetilde{w}_{k,i})_i$ such that $(\boldsymbol{W}_k\|\widetilde{\boldsymbol{W}}_k) = (\boldsymbol{w}_k\|\widetilde{\boldsymbol{w}}_k)G$, and that, because of the verified equations: $\sum_{i=1}^{N} w_{k,i} \cdot \boldsymbol{p}_i + \sum_{l=1}^{\widetilde{N}} \widetilde{w}_{k,l}\widetilde{\boldsymbol{p}}_l = \boldsymbol{c}_k$ So $\mathcal{C}$ has obtained: $\mathsf{m}_k \in \mathsf{pol}_{\mathcal{R}_t}(\{0;1\}^n), \mathsf{u}_k, \widetilde{\mathsf{u}}_k \in \mathsf{pol}_{\mathcal{R}_p}(\{0;1\}^n)$, $\mathsf{e}_k, \mathsf{e}'_k \in \mathsf{pol}_{\mathcal{R}_p}([\![2B]\!])$, and $\widetilde{\mathsf{e}}_k, \widetilde{\mathsf{e}}'_k \in \mathsf{pol}_{\mathcal{R}_p}([\![2^{\lambda+1}B]\!])$ such that:
$$(\mathsf{c}_k, \mathsf{c}'_k) = (\mathsf{p} \cdot (\mathsf{u}_k + \widetilde{\mathsf{u}}_k) + \Delta[\mathsf{m}_k]_t + \mathsf{e}_k + \widetilde{\mathsf{e}}_k, \mathsf{p}' \cdot (\mathsf{u}_k + \widetilde{\mathsf{u}}_k) + \mathsf{e}'_k + \widetilde{\mathsf{e}}'_k);$$

and the PQPKE decryption algorithm will thus return m_k on input $(\mathsf{c}_k, \mathsf{c}'_k)$ and the secret key corresponding to pk, sk. So now, instead of using sk to answer a valid query C_k, $\mathcal{C}$ uses the above extracted vectors to find and return m_k.

Under the perfect binding property of $\boldsymbol{\varphi}$, the m_k computed with this new method is identical to the one found using sk, so: $\Pr\{S_{\mathcal{A}}^{\mathbf{G}_7}(\lambda)\} = \Pr\{S_{\mathcal{A}}^{\mathbf{G}_6}(\lambda)\}$.

Game $\mathbf{G}_8$: is as the previous game, though now $\mathcal{C}$ is not provided with sk; as she was not using it, this does not change the distribution of reponses with respect to the previous game: $\Pr\{S_{\mathcal{A}}^{\mathbf{G}_8}(\lambda)\} = \Pr\{S_{\mathcal{A}}^{\mathbf{G}_7}(\lambda)\}$.

Game $\mathbf{G}_9$: is as the previous game, but now the challenger draws $\mathsf{b} \xleftarrow{\$} \mathcal{R}_p$ and sets: $(\mathsf{c}^*, \mathsf{c}^{'*}) \leftarrow (\mathsf{c}_b^*, \mathsf{c}_b^*) + (\mathsf{b} \cdot \mathsf{u}_1^* + \mathsf{e}_1^*, \mathsf{p}' \cdot \mathsf{u}_1^* + \mathsf{e}_1^{'*})$ inside of C^*. The adversary's advantage in distinguishing this game from the previous one is her advantage in the RLWE security game, challenged with either $(\mathsf{p}', \mathsf{b})$ or $(\mathsf{p}', -\mathsf{p}' \cdot \mathsf{s} + \mathsf{e})$, for p' drawn uniformly at random in $\mathcal{R}_p$, so:

$$\left|\Pr\{S_{\mathcal{A}}^{\mathbf{G}_9}(\lambda)\} - \Pr\{S_{\mathcal{A}}^{\mathbf{G}_8}(\lambda)\}\right| \leqslant \mathsf{Adv}_{\mathcal{A}}^{\mathsf{RLWE}}(\lambda).$$

Game $\mathbf{G}_{10}$: is as the previous game, though now, the challenger sets $(\mathsf{c}^*, \mathsf{c}^{'*}) \xleftarrow{\$} \mathcal{R}_p^2$. As u_1^* is in $\mathsf{pol}_{\mathcal{R}_p}(\{0;1\}^n \backslash \{\mathbf{0}_n\})$, which contains only invertible elements of $\mathcal{R}_p$, for each element $(\mathsf{x}, \mathsf{x}') \in \mathcal{R}_p^2$:

$$\begin{aligned}
&\Pr_{(\mathsf{a},\mathsf{b}) \xleftarrow{\$} \mathcal{R}_p^2} \left\{(\mathsf{c}_b^*, \mathsf{c}_b^*) + (\mathsf{b} \cdot \mathsf{u}_1^* + \mathsf{e}_1^*, \mathsf{a} \cdot \mathsf{u}_1^* + \mathsf{e}_1^{'*}) = (\mathsf{x}, \mathsf{x}')\right\} \\
&\quad = \Pr_{(\mathsf{a},\mathsf{b}) \xleftarrow{\$} \mathcal{R}_p^2} \left\{(\mathsf{c}_b^*, \mathsf{c}_b^*) + (\mathsf{e}_1^*, \mathsf{e}_1^{'*}) + (\mathsf{b}, \mathsf{a}) = (\mathsf{x}, \mathsf{x}')\right\} \\
&\quad = \Pr_{(\mathsf{a},\mathsf{b}) \xleftarrow{\$} \mathcal{R}_p^2} \left\{(\mathsf{b}, \mathsf{a}) = (\mathsf{x}, \mathsf{x}')\right\} = \frac{1}{|\mathcal{R}_p|^2} = p^{-2n}
\end{aligned}$$

So this game is perfectly indistinguishable from the previous one: $\Pr\{S_{\mathcal{A}}^{\mathbf{G}_{10}}(\lambda)\} = \Pr\{S_{\mathcal{A}}^{\mathbf{G}_9}(\lambda)\}$.

In this last game, all of $\mathcal{C}$'s answers are built without using sk, and are independent of b; therefore, $\mathcal{A}$'s advantage is null.

As a result: $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{TCCA}}(\lambda) \leqslant \mathsf{Adv}_{\mathcal{A}}^{\mathsf{coll},\mathcal{H}}(\lambda) + \frac{1}{p} + 3 \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathsf{SXDH}}(\lambda) + \mathsf{Adv}_{\mathcal{A}}^{\mathsf{RLWE}}(\lambda)$. □

B.4 Proof of the IND-CPA Security of TREnc

Proof. Let $\lambda \in \mathbb{N}$, $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a PPT adversary.

Game $\mathbf{G}_0$: is the original security game; with $(\mathsf{PK}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$, $\mathcal{A}_1$ is provided with PK, returns a state st and two messages $\boldsymbol{m}_0, \boldsymbol{m}_1 \in \{0;1\}^n$; the challenger draws $b \xleftarrow{\$} \{0;1\}$, sets $C_b \xleftarrow{\$} \mathsf{Enc}(\mathsf{PK}, \boldsymbol{m}_b)$, and returns $(\mathsf{st}, \mathcal{C}_b)$ to $\mathcal{A}_2$. $\mathcal{A}_2$ then returns b' and wins iff it is equal to b.

Game $\mathbf{G}_1$: is as the previous game, but now, the challenger keeps a simulation trapdoor when generating the Groth-Sahai CRS (with now none-public coins). The distribution of PK is not altered, so $\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_1}(\lambda) = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{CPA},\mathsf{TREnc}}(\lambda)$.

Game $\mathbf{G}_2$: is as the previous game, but now, using the Groth-Sahai CRS trapdoor, the challenger simulates the Groth-Sahai commitments and proofs. Under the perfect witness-indistinguishability of the CRS used: $\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_2}(\lambda) = \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_1}(\lambda)$.

Game $\mathbf{G}_3$: is as the previous game, but now the challenger, not receiving sk in the initialization, replaces the RLWE encryption to be sent to $\mathcal{A}_2$ to one replacing pk with $(\mathsf{p}', \mathsf{b})$, for $\mathsf{b} \xleftarrow{\$} \mathcal{R}_p$; the difference between this game and the previous one is an RLWE distinguisher, so: $\left|\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_3}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_2}(\lambda)\right| \leqslant \mathsf{Adv}_{\mathcal{A}}^{\mathsf{RLWE}}(\lambda)$.

Game $\mathbf{G}_4$: is as the previous game, except that now, the challenger takes a uniformly random element of $\mathcal{R}_p^2$ as the previous RLWE encryption to be returned to $\mathcal{A}_2$. As b acted as a perfect mask in the previous game, this is statistically indistinguishable: $\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_4}(\lambda) = \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_3}(\lambda)$. In this last game, what is returned to the adversary is completely independent from b, so her advantage is null.

Finally, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{CPA,TREnc}}(\lambda) \leqslant \mathsf{Adv}_{\mathcal{A}}^{\mathsf{RLWE}}(\lambda)$. □

B.5 Proof of Receipt-Freeness of the Voting Scheme

Proof (Receipt-Freeness). Let $\lambda \in \mathbb{N}$, and $\mathcal{A}$ be a PPT adversary.

Game $\mathbf{G}_0$: is the original receipt-freeness game, in which $\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_0}(\lambda) = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{RF}}(\lambda)$;

Game $\mathbf{G}_1$: is as the previous game, though now, even when $\beta = 0$, $\mathsf{SimSetupElection}$ is called instead of $\mathsf{SetupElection}$ to set up the keys, and the challenger keeps the simulation trapdoor τ issued from the simulated election setup generation. $|\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_1}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_0}(\lambda)| \leqslant \mathsf{Adv}_{\mathcal{A}}^{\mathsf{SimSetup}}\mathsf{Election}(\lambda)$.

Game $\mathbf{G}_2$: is as the previous game, though now, even when $\beta = 0$, the tally proofs returned to $\mathcal{A}$ from $\mathcal{O}\mathsf{tally}$ are generated using $\mathsf{SimProof}$ as in the $\beta = 1$ case. $\left|\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_2}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_1}(\lambda)\right| \leqslant Q_{\mathcal{O}\mathsf{tally}} \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathsf{SimProof}}(\lambda)$.

Game $\mathbf{G}_3$: is as the previous game, but now, when $\mathcal{A}$ calls the $\mathcal{O}\mathsf{receiptLR}$ oracle, the same b_1 is used for both ballot boxes (replacing the b_0 that was used with BB_0). Moreover, the secret key from the generation is not received by the challenger, who uses the decryption oracles from the TCCA security game when asked for decryptions (refusing the decryption of ballots sent to $\mathcal{O}\mathsf{receiptLR}$). The difference between this game and the previous one is $Q_{\mathcal{O}}\mathsf{receiptLR}$ times the TCCA security game, for $Q_{\mathcal{O}}\mathsf{receiptLR}$ the number of requests to $\mathcal{O}\mathsf{receiptLR}$ (using $Q_{\mathcal{O}}\mathsf{receiptLR}$ hybrid games with a TCCA difference to the previous one between the $\mathbf{G}_2$ and $\mathbf{G}_3$), and: $\left|\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_3}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_2}(\lambda)\right| \leqslant Q_{\mathcal{O}}\mathsf{receiptLR} \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathrm{TCCA}}(\lambda)$. In this last game, what $\mathcal{A}$ receives is totally independent of β, so $\mathsf{Adv}_{\mathcal{A}}^{\mathbf{G}_3}(\lambda) = 0$.

Finally,

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{RF}}(\lambda) \leqslant Q_{\mathcal{O}}\mathsf{receiptLR} \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathrm{TCCA}}(\lambda) + Q_{\mathcal{O}\mathsf{tally}} \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathsf{SimProof}}(\lambda) + \mathsf{Adv}_{\mathcal{A}}^{\mathsf{SimSetupElection}}(\lambda).$$

□