Practical Traceable Receipt-Free Encryption

Henri Devillez[1], Olivier Pereira[1,2], and Thomas Peters[1]

[1] UCLouvain – ICTEAM – Crypto Group, B-1348 Louvain-la-Neuve – Belgium
[2] Microsoft Research, Redmond, WA, USA
`first.last@uclouvain.be`

Abstract. Traceable Receipt-free Encryption (TREnc) is a verifiable public-key encryption primitive introduced at Asiacrypt 2022. A 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). This property can support applications like voting, and it was shown that receipt-free non-interactive voting, where voters are unable to convince any third party of the content of their vote, can be generically built from a TREnc.
While being a very promising primitive, the few existing TREnc mechanisms either require a secret coin CRS or are fairly demanding in time and space requirements. Their security proofs also come with a linear security degradation in the number of challenge ciphertexts.
We address these limitations and offer two efficient public coin TREnc mechanisms tailored for the two common tallying approaches in elections: homomorphic and mixnet-based. The TCCA security of our mechanisms also enjoys an almost-tight reduction to SXDH, based on a new randomizable technique of independent interest in the random oracle model.
A Rust implementation of our TREnc mechanisms demonstrates that we can verifiably encrypt 64 bits in less than a second, and full group elements in around 30 ms., which is sufficient for most real-world applications. While comparing with other solutions, we show that our approaches lead to the most efficient non-interactive receipt-free voting system to date.

1 Introduction

At Asiacrypt 2022, Devillez, Pereira and Peters [14] proposed a new verifiable public-key encryption primitive that they showed to be sufficient to build a non-interactive receipt-free voting system, generically. By the way, voters simply encrypt their votes through this new primitive and send the resulting ciphertext while keeping a trace of it. The ciphertext is processed by a server that is trusted for receipt-freeness (but not for privacy or integrity) and then posted on a public bulletin board, where the voter can verify its presence, if they wish so. That is, while the ciphertext processing by the server prevents any voter from proving to a third party how they voted, the ciphertext trace can convince the voter that their vote has not been altered. These two features of the encryption scheme led to the naming *Traceable Receipt-free Encryption*, or TREnc for short. More concretely, a TREnc enjoys mainly two additional features:

– *Traceability.* Each ciphertext comes with a public trace that is independent of the plaintext: as long as a voter keeps secret (or properly erases) the random coins used to compute the ciphertext, it will be infeasible for anyone, even with the secret decryption key, to produce a ciphertext encrypting a different message while having the same trace. This trace is what makes it possible for a voter to track his ballot through the voting system.
– *Traceable-CCA Security.* If an adversary chooses two ciphertexts with identical traces and is returned one of them after it has been randomized by a challenger, then the adversary cannot recognize which ciphertext was randomized, even with access to a decryption oracle (that refuses to decrypt ciphertexts with that same trace, but only on post-challenge query). This guarantees that, if a voter encrypts a vote, possibly in a malicious way, and submits it, and if the ciphertext is randomized before being posted on a bulletin board, then the voter becomes unable to demonstrate the content of that ciphertext to any third party.

Devillez et al. [14] also designed two constructions in the standard model to realize a TREnc under the SXDH assumption, i.e. under the DDH assumption in the bilinear groups endowed with pairings. However, the first construction is generic and relies on heavy public-key ingredients and no instantiation was specified. The second construction is fully instantiated and reasonably efficient but relies on a secret-key common reference string, which is challenging in practice and weakens the trust model required for elections. Moreover, while both schemes allow encrypting a full group element into a ciphertext, the authors left open the task of designing a practical public-coin TREnc, let alone one that allows to verifiably encrypt bits as needed in a voting system with homomorphic tally.

1.1 Our Contributions

We propose two new practical TREnc schemes that do not rely on any trusted setup and support the verifiable encryption of ℓ-bit strings for the first construction and the encryption of a full group element for the second. As a first step, we design these two public-coin schemes in the standard model from pairings and prove their security under the SXDH assumption. In particular, the TCCA security holds in the original (single-challenge) definition. In parallel, we observe that the proof of the generic transformation of Devillez [14] of a TREnc into a receipt-free voting system implicitly relies on a tight reduction to a multi-challenge extension of the TCCA definition, and that the TCCA notion implies its multi-challenge variant with a security loss proportional to the number of challenges. We formally state the multi-challenge notion and the implication here. As a second step, we bring a very slight modification to our constructions by turning them in the random oracle model (ROM), which allows us to prove the multi-challenge TCCA notion with tighter reduction to SXDH. More precisely, we have an almost-tight security for the first scheme, where the security loss is proportional to the number ℓ of bits per ciphertext and is essentially independent of the adversary and its number of challenge, decryption and hash queries. The proof of the second construction encrypting group elements is tight.

As an implication, we get two practical voting schemes that are tailored for the two common tallying approaches in elections: the homomorphic approach where encrypted votes are aggregated into an encrypted tally that is decrypted, and the mixnet-based approach where encrypted votes are going through a mixnet before being decrypted. Furthermore, the receipt-freeness of these schemes (almost) tightly relates to SXDH. Moreover, since ℓ is usually selected independently of the security parameter in a voting system, this further shortens the security loss.

Eventually, we implement our TREnc mechanisms in Rust to demonstrate that we can verifiably encrypt 64 bits in less than a second and a full group element in around 30 ms., which is sufficient for most real-world applications. In our comparison section, we also provide benchmarks also showing that our design are more than twice as fast and 33% shorter than the most efficient alternative to build a non-interactive receipt-free voting scheme without a TREnc.

1.2 Other related works

The first design of a receipt-free voting protocol that relies on a randomizing server that is only trusted for receipt-freeness comes from Hirt and Sako [19]. Hirt's construction requires interaction between the voter and the server, which can be cumbersome. More than 10 years later, Blazy et al. introduced the concept of signatures on randomizable ciphertexts [9], which opened the way to non-interactive receipt-free voting. Non-interactive receipt-free voting was then formalized and a full construction was proposed by Chaidos et al. in BeleniosRF [10]. The BeleniosRF solution supports bit-by-bit encryption and encrypts ℓ bits into a ciphertext of size dominated by 6ℓ group elements in the first and second pairing groups. Our TREnc mechanism tailored for the encryption of bits has a ciphertext size dominated by 4ℓ group elements in the first and second groups. Moreover, the security analysis of the BeleniosRF encryption mechanism has a simulated decryption complexity that is exponential in the number of encrypted bits, which makes it poorly adapted to secure expressive ballots that may contain hundreds of voter selections and are traditionally tallied using a mixnet.

Recently, Doan et al. [15] proposed a novel public-coin TREnc to verifiably encrypt one bit under the SXDH assumption. However, the goal of this scheme was to show how a TREnc can also be designed to offer another security property valuable in election called perfectly-private audit trail. In that case, only a perfectly hiding portion of a ciphertext is made available on the bulletin board, in a way that keeps all the other security notions, and the recept-freeness. Achieving both properties together actually comes with additional elements in the ciphertext – around 100 group elements for encrypting a single bit – which makes the scheme more of a conceptual interest, very far from the efficiency goals of the constructions presented here.

Recently, Pointcheval designed new efficient NIZK of subset membership for proving vote validity and demonstrates their compatibility with TREnc mechanisms [22]. The benefits of TREnc mechanisms have also been further explored in the context of the cast-as-intended verifiability property [13].

1.3 Overview and Techniques

In both constructions, our ciphertexts have the following structure

$$\mathsf{CT} = (\mathbf{c}, \mathbf{ovk}, \boldsymbol{\sigma}_{\mathsf{trace}}, \boldsymbol{\pi}_{\mathsf{valid}}),$$

where $\mathbf{c}$ is a chosen-plaintext (CPA) secure encryption of the message, $\mathbf{ovk}$ is a one-time verification key of a linearly homomorphic structure-preserving signature (LHSP) scheme [21] that is freshly generated at the encryption time and constitutes the trace of the ciphertext, $\boldsymbol{\sigma}_{\mathsf{trace}}$ is the component offering traceability, and $\boldsymbol{\pi}_{\mathsf{valid}}$ is a tag-based randomizable proof with associated tag $\mathbf{ovk}$ and that ensures that the CPA part $\mathbf{c}$ is well-formed.

Below, we give more insight about how these ciphertext parts are designed. We first start with the construction encrypting ℓ-bit string and then move to the other that encrypts a group element. In both cases, we begin with the description in the standard model before turning to the tighter ROM-based variant.

Encrypting ℓ bits To encrypt an ℓ-bit message $\boldsymbol{m} = (m_1, \ldots, m_\ell)$, we rely on a homomorphic ElGamal-like encryption $\mathbf{c} = (d_1, d_2, c_1, \ldots, c_\ell) \in \mathbb{G}^{\ell+2}$ such that

$$\mathbf{c} = (1, 1, g_1^{m_1}, \ldots, g_\ell^{m_\ell}) \cdot (g, h, f_1, \ldots, f_\ell)^\theta$$

with a single random coin θ, where $f_i = g^{\alpha_i} h^{\beta_i}$ and g_i are part of the public key, for all $1 \leq i \leq \ell$. The secret key $\mathsf{SK} = (\mathsf{sk}_i)_{i=1}^{\ell} = (\alpha_1, \beta_1, \ldots, \alpha_\ell, \beta_\ell)$ allows extracting $g_1^{m_1}, \ldots, g_\ell^{m_\ell}$. Reusing θ saves a linear term in ℓ with respect to ℓ independent ElGamal encryptions of the bits m_i.

Public-coin construction in the standard model. The tracing part $\boldsymbol{\sigma}_{\mathsf{trace}}$ prevents the malleability of the message-carrying components $g_1^{m_1}, \ldots, g_\ell^{m_\ell}$ while keeping the randomizability of the ciphertext $\mathbf{c}$. It relies on LHSP signatures on the vectors $(g, c_1, \ldots, c_\ell, d_1)$ from the CPA encryption and $(1, f_1, \ldots, f_\ell, g)$ for randomizing the CPA part that allow deriving signatures on any linear combination. However, we restrict the validity on vectors that start with the component g to ensure that the underlying $(g, g_1^{m_1}, \ldots, g_\ell^{m_\ell}, 1)$ remains unchanged after randomization. We excluded d_2 and h from the elements signed with the LHSP signature scheme because $\boldsymbol{\pi}_{\mathsf{valid}}$ will ensure that $(d_1, d_2) = (g, h)^\theta$ as well as any of its randomization; there is no need to further enforces d_2 to follow the same randomization via the LHSP signatures. To reduce the linear size of $\boldsymbol{\sigma}_{\mathsf{trace}}$ in ℓ to a constant, we actually shrink these vectors as (g, c, d_1) and $(1, f, g)$, where $c = \prod_{i=1}^{\ell} c_i$ and $f = \prod_{i=1}^{\ell} f_i$, which still allows randomizing $\mathbf{c}$ and adapting its LHSP signature. Since $c = g_1^{m_1} \cdots g_\ell^{m_\ell} f^\theta$ can be seen as a binding commitment, we can combine the unforgeability with the hardness of computing discrete logarithms to keep the traceability. While this strategy seems enough for traceability, like [14] we have to sign a third vector that will help randomizing with an SXDH instance in the TCCA proof. That is because the TCCA notion requires the challenger to reuse the same trace $\mathbf{ovk}$ of the adversary, while ignoring the corresponding one-time signing key $\mathbf{osk}$. Another solution would have been to add a proof of knowledge of $\mathbf{osk}$ with an online extractor, but it would be very expensive. In summary, up to additional technical modifications, $\boldsymbol{\sigma}_{\mathsf{trace}}$ authenticates the row space of the matrix

$$\mathbf{T} = \begin{pmatrix} g & c & d_1 \\ 1 & f & g \\ 1 & F & G \end{pmatrix}, \tag{1}$$

where F and G are uniform elements in $\mathbb{G}$ added to the public-key PK. Now, we turn to the randomizable tag-based proof $\boldsymbol{\pi}_{\mathsf{valid}}$ that $\boldsymbol{c}$ is well-formed. To ensure that each triple (d_1, d_2, c_i) for $i \in [\ell]$ is indeed an encryption of a bit m_i, we not only need a randomizable proof, but a malleable proof that can be adapted when the word $\mathbf{c}$ in the language is modified through randomization. In pairing groups, we can thus follow two directions: either the Groth-Sahai (GS) proof system system [18] or Couteau and Hartmann (CH) compiler [12] of Σ-protocols even if this property in less commonly known for the latter. For our needs, it happens that both choices lead to the same efficiency under SXDH (as we need them to be partially extractable), and both are public coin. For simplicity, we rely on GS proofs to ensure that each triple (d_1, d_2, c_i) for $i \in [\ell]$ is indeed an encryption of a bit m_i by showing that $m_i(1 - m_i) = 0$ in the field of scalars in the exponents. To optimize the proof, and unlike [10, 15], we avoid relying on quadratic multi-scalar multiplication equations or quadratic pairing product equations (in the terminology of GS) that respectively have proof sizes of 6ℓ and 8ℓ group elements in both source groups. Instead, we "linearize" the multi-scalar multiplication equation by observing that it is enough to satisfy

$$
\begin{aligned}
d_1 &= g^{\theta} & d_2 &= h^{\theta} & c_i &= g_i^{m_i} f_i^{\theta} & \quad (2)\\
&= d^{m_i} g^{\vartheta_i} & & & &= c_i^{m_i} f_i^{\vartheta_i}, & \quad (3)
\end{aligned}
$$

where $\vartheta_i = \theta(1 - m_i)$. Intuitively, Equation 2 implies that the ciphertext (d_1, d_2, c_i) encrypts some message $m_i \in \mathbb{Z}_q$, and Equation 3 then implies that $(d_1, c_i)^{(1-m_i)}$ is an encryption of 0.[3] This representation leads to proofs of size 4ℓ group elements in both source groups. However, we do not only need zero-knowledge proof of knowledge of the m_i's. We actually need a tag-based simulation-sound extractable proof, where simulating a proof for a false statement does not help the adversary to produce valid proofs for which we cannot compute the witness $(m_i)_{i=1}^{\ell} \in \{0, 1\}^{\ell}$. To build such a proof, we rely on the well-known OR-proof technique, and show that either the statement is true or that "I know an opening of a perfectly hiding commitment C", where the commitment is derived from the tag $\tau \leftarrow \mathsf{Hash}(\mathbf{ovk})$ as $C = C_0^{\tau} C_1$ for uniform elements C_0 and C_1 of $\mathbb{G}$ included in PK and seen as Pedersen commitments. If the adversary manages to prove the simulated branch, i.e. the knowledge of an opening, the challenger will be able to extract another opening than the one it programs in the public key, which contradicts the binding property. To remain randomizable, we implement this OR-proof by following a technique due to Rafols [23] which first computes two correlated common reference strings (CRS) so that at most one allows simulating GS proofs.[4] We observe that it is indeed possible to also randomize the CRSes. The resulting proof system allows us to answer decryption queries even for $C = C_0^{\tau} C_1$ that will occur in the challenge query as long as the decryption query associated to $\mathbf{ovk}$ happens before. Nevertheless, we have to bring a last modification to this proof system to switch between the branches being proved during the computation of the randomized challenge ciphertexts in the TCCA security game. That is because the given ciphertexts must use the "real" branch while by randomizing with the SXDH instance related to (F, G) we will push the word outside the language, which forces us to use the "simulated branch." For technical reason, we need to add another statement in parallel of C for which we control, during the transition of the games, whether proving the simulated branch is feasible at all or not, which allows us to implicitly control the distribution of the correlated CRSes (when the reduction must still does not know any trapdoor information). This last step is crucial to proving the TCCA security and it is made clear in the intuition and in the security proof (Section 6).

Almost-Tightness in the ROM. The concept of tight multi-challenge CCA encryption (mCCA) was put forth by Bellare, Boldyreva and Micali [6]. Thanks to the random-self reducibility of DDH (and thus SXDH) it is easy to get a tightly-secure publicly-verifiable mCCA encryption by combining this technique with Gennaro-Shoup [24]. Indeed, the special soundness of the underlying Σ-protocol is enough to have a tight unbounded simulation-sound argument with the Fiat-Shamir transform without rewinding. However, achieving tight TCCA TREnc is not that straightforward. As argued above, and independently of tightness, achieving TCCA in the single challenge setting already requires to avoid partitioning the tags derived from the traces between

[3] This is why we can also rely on CH compiler as there is a natural Σ-protocol to show this statement.
[4] We can still rely on CH to have the same property with the same size.

the adversary and the challenger: the challenger must use the traces generated by the adversary, and their related proofs must remain extractable when computed by the adversary, and simulatable when *randomized* by the challenger. Therefore, the random oracle cannot be programmed differently for the adversary and the challenger. Moreover, even if we rely on the ROM, our TREnc still needs the randomizability property and the adaptability of the proof $\boldsymbol{\sigma}_{\mathsf{trace}}$. Since the Fiat-Shamir transform fixes the challenge and that we also need a practical online extractable proof, it is not compatible with the TREnc features. Therefore, we almost keep the whole above proof system we just described in the standard model but modify the computation of the tag-based commitment to allow injecting fresh values of C for each new value of the trace $\mathsf{\textbf{ovk}}$. This makes it possible to refresh the argument on the openings and the binding property on any challenge queries since this is where the simulation-sound extractability emerges. This was the easy step related to the proof that $\mathbf{c}$ is well-formed. Now, we turn to the randomization of those $\mathbf{c}$ that should become independent of the choice made by the challenger in all the challenge queries of the TCCA game in the multi-challenge setting.[5] The issue if we stick with the standard model description lies in the use of the last row of the matrix $\mathbf{T}$ in Equation 1. Even if we are able to use $(1, F, G)$ to "over-randomize" (g, c, d_1) in all the challenge queries, we must over-randomize all the c_i individually and through all the queries. To rely on an information theoretic argument to ensure that all the randomized challenge ciphertexts are made independent of each other, we need a fresh row per ciphertext with uniform F, G. Therefore, we derive from the trace $(C, F, G) \leftarrow \mathsf{Hash}(\mathsf{\textbf{ovk}})$, where C is the tag-based commitment described above, and (F, G) is fresh for every new trace as we need. With a carefull analysis, we manage to show that we can also rely on the random self reducibility to control these rows together per index $i \in [\ell]$.

Encrypting a group element To encrypt a group element $m \in \mathbb{G}$, we compute the CPA encryption part as $\mathbf{c} = (d_1, d_2, d_3) = (1, 1, m) \cdot (g, h, f)^{\theta}$, where the secret key $\mathsf{SK} = (\alpha, \beta)$ satisfies $f = g^{\alpha} h^{\beta}$. We could simply replace $g_1^{m_1}$ by m in the first construction, set $\ell = 1$, and remove from $\boldsymbol{\pi}_{\mathsf{valid}}$ the parts related to the satisfiability of bits. This would enjoy all the properties, but we can shorten the ciphertext size by relying on another idea.

Public-coin construction in the standard model. The traceability part $\boldsymbol{\sigma}_{\mathsf{trace}}$ is like in the first construction, but without compression since we have a single message-carrying part d_3. So, let us focus on the validity proof. Here, we adapt the CH technique to be compatible with the adaptive tag and the simulation technique by randomization. We can make a proof for witness-samplable parameter (g, h) as we manage to rely on (simulation) soundness only at the end of all our transitions in the TCCA proof, and especially after the SXDH transition involving a uniform instance (g, h, G, H) to "over-randomize" $\mathbf{c}$ also with $f = g^{\alpha} h^{\beta}$ and $F = G^{\alpha} H^{\beta}$. Therefore, we can start with the most efficient CH proof of the form $(X_1, X_2, \hat{Z})$ based on a Σ-protocol, where (X_1, X_2) relates to the commit phase (i.e., the first flow) and $Z \in \hat{\mathbb{G}}$ relates to the response phase (i.e., the third flow), and with a uniform element $E \in \hat{\mathbb{G}}$ encoding the (fixed) challenge (i.e. the second flow) available in the CRS of the system so that $e(d_1, E) \cdot e(X_1, \hat{g}) = e(g, \hat{Z})$ and $e(d_2, E) \cdot e(X_2, \hat{g}) = e(h, \hat{Z})$. Even if we do not use an OR-proof technique anymore, we still need simulation by randomization as with the perfectly hiding commitment of the first construction so as to also ensure that the adversary cannot simulate proofs given a simulated proof of a false statement. However, CH proof only provides soundness. Nevertheless, by somehow embedding this principle into E, the reduction can represent it as a Pedersen commitment $E = \hat{g}^x \hat{h}^y$ with (x, y) as the simulation key. Therefore, we include uniform $E_0 = \hat{g}^{x_0} \hat{h}^{y_0}$ and $E_1 = \hat{g}^{x_1} \hat{h}^{y_1}$ in PK, and only compute $E = E_0^{\tau} E_1$ in the proof with $\tau \leftarrow \mathsf{Hash}(\mathsf{\textbf{ovk}})$. Now, a valid proof $(X_1, X_2, Y_1, Y_2, \hat{Z})$ satisfies

$$e(d_1, E) \cdot e(X_1, \hat{g}) \cdot e(Y_1, \hat{h}) = e(g, \hat{Z})$$
$$e(d_2, E) \cdot e(X_2, \hat{g}) \cdot e(Y_2, \hat{h}) = e(h, \hat{Z})$$

and we get all what we want at the cost of only two additional elements. To show the one-time simulation-soundness, we rely on the trapdoor membership $\omega = \log_g h$ that we are free to use after the SXDH transition

[5] We extend the TCCA notion in the multi-challenge setting. It is also easy to extend it in the multi-user setting. For simplicity, we do not focus on the latter as it is anyway easy to achieve from the random self reducibility of SXDH.

related to (g, h). First, we note that in the pre-challenge phase any E perfectly hides the simulation key, and in the post-challenge phase the same holds for any $E \neq E^*$ with $\tau^* \neq \tau \leftarrow \mathsf{Hash}(\mathbf{ovk})$. That is, even if simulation is possible for an unbounded adversary, it will use another simulation key than the challenger, which is enough to distinguish a (last) SXDH instance related to $(\hat{g}, \hat{h})$ as long as the word (d_1, d_2) is not in the language. Indeed, if $d_1^\omega \neq d_2$ it would lead to the non trivial equation $e(d_1^\omega/d_2, E) \cdot e(X_1^\omega/X_2, \hat{g}) \cdot e(Y_1^\omega/Y_2, \hat{h}) = 1_{\mathbb{G}_T}$ and the challenger will have its own based on the corresponding (hidden) (x, y), will leads to the non-trivial relation $e(A, \hat{g}) \cdot e(B, \hat{h}) = 1_{\mathbb{G}_T}$.

Tightness in the ROM. For this part we can simply follows the technique described for the first construction as ℓ=1, but with $(E, F, G) \leftarrow \mathsf{Hash}(\mathbf{ovk})$.

1.4 Paper Roadmap

We organize the paper as follows. Section 2 provides the cryptographic building blocks of our constructions. Section 3 describes our new TREnc mechanisms and Section 4 describes a more efficient variant of the construction tailored to Mixnet voting. In Section 5, we detail our implementations and compare our performance with the state of the art, and we conclude in Section 6 by recalling the voting application.

2 Building blocks

We recall the syntax and the security definitions of the cryptographic primitive of TREnc [14] before turning to other cryptographic schemes that will be helpful for our new TREnc constructions with, unlike [14], public-coin parameters.

2.1 Traceable Receipt-Free Encryption

Definition 1 (TREnc). *A* Traceable Receipt-Free Encryption *scheme is a public-key encryption scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *augmented with a 5-tuple of algorithms* $(\mathsf{LGen}, \mathsf{LEnc}, \mathsf{Trace}, \mathsf{Rand}, \mathsf{Ver})$ *where* $\mathsf{Enc}(\mathsf{pk}, m) = \mathsf{LEnc}(\mathsf{pk}, \mathsf{LGen}(\mathsf{pk}), m)$*, and:*

– $\mathsf{LGen}(\mathsf{pk}; r)$*: The link generation algorithm takes as input a public encryption key* pk *in the range of* Gen *and randomness* r*, and outputs a link key* lk*.*
– $\mathsf{LEnc}(\mathsf{pk}, \mathsf{lk}, m; r)$*: The linked encryption algorithm takes as input a pair of public/link keys* $(\mathsf{pk}, \mathsf{lk})$*, a message* m *and randomness* r *and outputs a ciphertext.*
– $\mathsf{Trace}(\mathsf{pk}, c)$ *: The tracing algorithm takes as input a public key* pk*, a ciphertext* c *and outputs a trace* t*. We call* t *the trace of* c*.*
– $\mathsf{Rand}(\mathsf{pk}, c; r)$*: The randomization algorithm takes as input a public key* pk*, a ciphertext* c *and randomness* r *and outputs another ciphertext.*
– $\mathsf{Ver}(\mathsf{pk}, c)$*: The verification algorithm takes as input a public key* pk*, a ciphertext* c *and outputs* 1 *if the ciphertext is valid,* 0 *otherwise.*

Sometimes, we omit the randomness r *from the notations. By abusing notation,* $c \in \mathsf{Enc}(\mathsf{pk})$ *means that the ciphertext* c *belongs to the range of honestly generated encryptions of some message with* pk*, and* $c \in \mathsf{Enc}(\mathsf{pk}, m)$ *means that the ciphertext* c *belongs to the range of honestly generated encryptions of* m *with* pk*.*

Correctness Beyond the usual correctness of encryption scheme, a TREnc must satisfy:

– *Link traceability:* For every pk in the range of Gen, every lk in the range of $\mathsf{LGen}(\mathsf{pk})$, and every pair of messages (m_0, m_1), the following equality holds: $\mathsf{Trace}(\mathsf{pk}, \mathsf{LEnc}(\mathsf{pk}, \mathsf{lk}, m_0)) = \mathsf{Trace}(\mathsf{pk}, \mathsf{LEnc}(\mathsf{pk}, \mathsf{lk}, m_1))$.
– *Honest verifiability:* For every pk in the range of Gen and every message m, it holds that $\mathsf{Ver}(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, m)) = 1$.

– *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 $\mathsf{Trace}(\mathsf{pk}, c) = \mathsf{Trace}(\mathsf{pk}, \mathsf{Rand}(\mathsf{pk}, c))$.

We note that the link traceability property ensures that the trace of a ciphertext does not depend on the message that is encrypted: the link key is (fortunately) not a receipt of a ciphertext.

Security notions A TREnc must be verifiable, strongly randomizable, traceable and TCCA-secure, as defined next. The first three notions hold even when the adversary is given the secret key of the TREnc so that malicious authorities cannot abuse the users. The last notion of TCCA is a privacy notion that holds even if the user is malicious.

Definition 2 (Verifiability). *A TREnc is* verifiable *if for every efficient adversary $\mathcal{A}$,* $\Pr[\mathsf{Ver}(\mathsf{pk}, c) = 1$ *and* $c \notin \mathsf{Enc}(\mathsf{pk}, \cdot) | (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda); c \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{sk})]$ *is negligible in λ.*

Definition 3 (Strong Randomization). *A TREnc is* strongly randomizable *if for every $c \in \mathsf{LEnc}(\mathsf{pk}, \mathsf{lk}, m)$ with pk from $(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{Gen}(1^\lambda)$ and lk in the range of $\mathsf{LGen}(\mathsf{pk})$, the following computational indistinguishability relation holds:*

$$\mathsf{Rand}(\mathsf{pk}, c) \approx_c \mathsf{LEnc}(\mathsf{pk}, \mathsf{lk}, m)$$

Together with the verifiability notion, this definition tells that a re-randomized valid ciphertext is still valid, and then belongs to the range of honest ciphertexts. The publicly traceable randomization from the correctness then implies that this re-randomized ciphertext decrypt to the same message.

The next definition ensures that valid ciphertexts that trace to each other contain the same message as long as a link key is only used once.

Definition 4 (Traceability). *A TREnc is* traceable *if for every efficient adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the experiment $\mathsf{Exp}^{\mathrm{trace}}_{\mathcal{A}}(\lambda)$ in Figure 1 (right) returns 1 with a probability negligible in λ.*

So, even if TREnc ciphertexts are malleable, the traceability implies the non-malleability of the plaintexts against malicious authorities with access to sk.

The TCCA security is a CCA-like security notion that differs from all existing CCA-like notions by letting the adversary submit pairs of ciphertexts instead of pairs of messages. It guarantees that a randomized ciphertext becomes indistinguishable from any other ciphertext that has the same trace. Furthermore, we know from the link traceability that the encryption of any vote could have that trace. This essentially guarantees the absence of a vote receipt, even if voters bias the ciphertexts they send.

We introduce here the notion of mTCCA security, in which the adversary can make a polynomial number of challenge queries. It can be observed that, in this definition, the adversary can always request decryption on any ciphertext as long as its trace was not already involved in a challenge query. That is, decryption is allowed for any trace even if they later occur in a challenge query. This property was crucial in the original definition to allow reaching receipt-freeness when a TREnc is used in a voting system because the secret key always make it possible to compute (or simulate) the result of the election. Our new experiment respects this fundamental feature.

Definition 5 (mTCCA). *A TREnc is secure against traceable chosen-ciphertext attacks in the multi-challenge setting if for every efficient adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ the experiment $\mathsf{Exp}^{\mathrm{m-tcca}}_{\mathcal{A}}(\lambda)$ defined in Figure 1 (left) returns 1 with a probability negligibly close in λ to $\frac{1}{2}$.*

Obviously, the mTCCA notion implies the TCCA notion. Conversely, we state that new mTCCA notion is implied by the the original TCCA notion of [14].

Lemma 1. *If a TREnc is TCCA, then it is mTCCA. More precisely, for any adversary $\mathcal{A}$ with success probability ε^*_{tcca} in the mTCCA game and making q challenge queries, we have $|\varepsilon^*_{tcca} - 1/2| \leq q \cdot |\varepsilon_{tcca} - 1/2|$, where ε_{tcca} is the maximal success probability of any adversary in the TCCA experiment.*

```
$\mathsf{Exp}^{\mathrm{m-tcca}}_{\mathcal{A}}(\lambda)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{Gen}(1^\lambda)$
$b \leftarrow\$ \{0, 1\}$
$L \leftarrow \{\}$
$b' \leftarrow\$ \mathcal{A}^{\mathsf{Dec}(\cdot),\mathsf{Chall}(\cdot)}(\mathsf{pk})$
return $b' = b$
```

```
$\mathsf{Chall}(c_0, c_1)$
if $\mathsf{Trace}(c_0) \in L$ or
$\mathsf{Trace}(\mathsf{pk}, c_0) \neq \mathsf{Trace}(\mathsf{pk}, c_1)$or
$\mathsf{Ver}(\mathsf{pk}, c_0) = 0$ or $\mathsf{Ver}(\mathsf{pk}, c_1) = 0$ :
    return $\perp$
else $L \leftarrow L \cup \{\mathsf{Trace}(c_b)\}$
return $\mathsf{Rand}(\mathsf{pk}, c_b)$
```

```
$\mathsf{Dec}(c)$
if $\mathsf{Trace}(c) \in L$ : return $\perp$
else return $\mathsf{Dec}(\mathsf{sk}, c)$
```

```
$\mathsf{Exp}^{\mathrm{trace}}_{\mathcal{A}}(\lambda)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{Gen}(1^\lambda)$
$(m, \mathsf{st}) \leftarrow\$ \mathcal{A}_1(\mathsf{pk}, \mathsf{sk})$
$c \leftarrow\$ \mathsf{Enc}(\mathsf{pk}, m)$
$c^\star \leftarrow\$ \mathcal{A}_2(c, \mathsf{st})$
if $\mathsf{Trace}(\mathsf{pk}, c) = \mathsf{Trace}(\mathsf{pk}, c^\star)$ and
$\mathsf{Ver}(\mathsf{pk}, c^\star) = 1$ and $\mathsf{Dec}(\mathsf{sk}, c^\star) \neq m$
then return 1
else return 0
```

Fig. 1: mTCCA and trace experiments.

Proof (Proof of Lemma 1). Let q be the total number of challenge queries made by an adversary $\mathcal{A}$ in the multi-challenge TCCA experiment. We build a sequence of $q+1$ hybrids. In the i-th hybrid, we randomize c_1 in the first i challenges and randomize c_0 in the last $q-i$ challenges. For $i = 0$, $\mathcal{A}$ is in the mTCCA experiment when the random bit is 0. For $i = q$, $\mathcal{A}$ is in the mTCCA experiment when the random bit is 1. In the multi-challenge TCCA experiment, the adversary $\mathcal{A}$ will not notice the difference between the first and the last hybrid as proven next.

We reduce each transition to the single-challenge TCCA experiment. Let $i \in [q]$. Given $\mathcal{A}$, we build an adversary $\mathcal{B}$ emulating the transition between the $i-1$-th and i-th hybrid from the its TCCA experiment. The adversary $\mathcal{B}$ can query its own decryption oracle to answer these analogue queries made by $\mathcal{A}$. However, if $\mathcal{A}$ tries to get the decryption of a ciphertext that contains a trace for which $\mathcal{B}$ already emulated the answer to a j-th challenge query for some $j \neq i$, $\mathcal{B}$ returns $\perp$ and does not call its decryption oracle.

For each j-th challenge query with $j < i$, $\mathcal{B}$ checks if the pair of ciphertexts given by $\mathcal{A}$ are valid, if they share the same trace, and if this trace was not already involved in a previous challenge query. If not, $\mathcal{B}$ outputs $\perp$. Otherwise, it simply randomizes the second ciphertext of the pair and returns it to $\mathcal{A}$. If $j > i$, $\mathcal{B}$ does the same except that it randomizes the first ciphertext of the pair if it does not return $\perp$. For the i-th query, $\mathcal{B}$ simply calls its own challenge oracle and return the received output to $\mathcal{A}$.

If the hidden bit is 0, $\mathcal{B}$ emulates the $(i-1)$-th hybrid. In contrary, if the hidden bit is 1, $\mathcal{B}$ emulates the i-th hybrid. If we let ε_i be $\mathcal{A}$'s probability to rightly guess the hidden bit in the above i-th game, we clearly have $|\varepsilon_i - 1/2| \leq |\varepsilon_{\mathtt{tcca}} - 1/2|$, where $\varepsilon_{\mathtt{tcca}}$ it the maximal probability that $\mathcal{B}$ succeeds in guessing the right bit in the single-challenge TCCA game. Now, if $\varepsilon^*_{\mathtt{tcca}}$ is $\mathcal{A}$'s success probability in the multi-challenge TCCE game, we clearly have $|\varepsilon^*_{\mathtt{tcca}} - 1/2| \leq q \cdot |\varepsilon_{\mathtt{tcca}} - 1/2|$. $\square$

2.2 Computational setting

Given a security parameter λ, Setup is an efficient algorithm that generates common public parameters such that $\mathsf{Setup}(1^\lambda)$ outputs a bilinear group $\mathsf{pp} = (\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, p, e, g, \hat{g})$ with $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T$ of prime order $p > 2^{\mathrm{poly}(\lambda)}$ for some polynomial poly, where $g \leftarrow\$ \mathbb{G}$ and $\hat{g} \leftarrow\$ \hat{\mathbb{G}}$ are generators and $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$ is a bilinear map. In this setting, we rely on the SXDH assumption, which states that the DDH problem is hard in both $\mathbb{G}$ and $\hat{\mathbb{G}}$.

Following the common Groth-Sahai notation, we define the map $\iota : \mathbb{G} \to \mathbb{G}^2$ that maps $X \in \mathbb{G}$ to $\iota(X) = (X, 1)$ and the map $\iota_T : \mathbb{G}_T \to {\mathbb{G}_T}^2$ that maps $T \in \mathbb{G}_T$ to $\iota_T(T) = (T, 1)$. We also extend the pairing as $E_1 : \mathbb{G}^2 \times \hat{\mathbb{G}} \to \mathbb{G}_T^2$ such that $E_1(\boldsymbol{a}, b) = (e(a_1, b), e(a_2, b))$ and as $E_2 : \mathbb{G} \times \hat{\mathbb{G}}^2 \to \mathbb{G}_T^2$ such that $E_2(a, \boldsymbol{b}) = (e(a, b_1), e(a, b_2))$, where $\boldsymbol{a} = (a_1, a_2)$ and $\boldsymbol{b} = (b_1, b_2)$. We use the multiplicative notation for vector space operations.

2.3 Linearly Homomorphic Structure-Preserving Signatures

A central tool for our efficient TREnc construction is linearly homomorphic structure-preserving (LHSP) signatures. The structure preserving [4, 5] property makes it possible to sign messages that are vectors of group elements. In our case, these elements will be components of an encrypted vote intent. The additional linearly homomorphic feature, introduced by Libert et al. [21], allows deriving a signature on any linear combination (in the exponents) of already signed vectors so that its validity attests of its linear membership in the related linear span. In the voting protocol, it will be possible to re-randomize the ciphertext and adapt its signature while carefully guaranteeing the non-malleability of the plaintext.

Here, we only recall the *one-time* LHSP signature scheme of [21] (expressed in the SXDH setting as in [20]) as the voter will use the secret signing key to authenticate a single encryption/subspace. Below, $\hat{h} \leftarrow\!\!\$\, \hat{\mathbb{G}}$ is added to $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$.

Keygen(pp, n): given the public parameter pp and the (polynomial) space dimension $n \in \mathbb{N}$, choose $\chi_i, \gamma_i \leftarrow\!\!\$\, \mathbb{Z}_p$ and compute $\hat{g}_i = \hat{g}^{\chi_i}\hat{h}^{\gamma_i}$, for $i = 1$ to n. The private key is $\mathsf{sk} = \{(\chi_i, \gamma_i)\}_{i=1}^n$ and the public key is $\mathsf{pk} = \{\hat{g}_i\}_{i=1}^n \in \mathbb{G}^n$.

Sign(sk, $(M_1, \ldots, M_n)$): to sign a vector $(M_1, \ldots, M_n) \in \mathbb{G}^n$ using $\mathsf{sk} = \{(\chi_i, \gamma_i)\}_{i=1}^n$, output $\sigma = (Z, R) = \big(\prod_{i=1}^n M_i^{\chi_i}, \prod_{i=1}^n, M_i^{\gamma_i}\big)$.

SignDerive(pk, $\{(\omega_j, \sigma_j)\}_{i=1}^{\ell}$): given pk as well as ℓ tuples (ω_j, σ_j), parse σ_j as $\sigma_j = \big(Z_j, R_j\big)$ for $j = 1$ to ℓ. Return the triple $\sigma = (Z, R) \in \mathbb{G}$, where $Z = \prod_{j=1}^{\ell} Z_j^{\omega_j}$, $R = \prod_{j=1}^{\ell} R_j^{\omega_j}$. That is, $\sigma = \prod_{j=1}^{\ell} \sigma_j^{\omega_j} \in \mathbb{G}^2$. *Note: if σ_j is a signature on $\boldsymbol{V_j} \in \mathbb{G}^n$, for $j = 1$ to ℓ, then σ is a signature on $\boldsymbol{V} = \prod_{j=1}^{\ell} \boldsymbol{V_j}^{\omega_j}$.*

Verify(pk, σ, $(M_1, \ldots, M_n)$): given $\sigma = (Z, R) \in \mathbb{G}^2$ and $(M_1, \ldots, M_n)$, return 1 if and only if $(M_1, \ldots, M_n) \neq (1_{\mathbb{G}}, \ldots, 1_{\mathbb{G}})$ and (Z, R) satisfies

$$e(Z, \hat{g}) \cdot e(R, \hat{h}) = \prod_{i=1}^{n} e(M_i, \hat{g}_i) \ . \tag{4}$$

2.4 Groth-Sahai proof system

Groth-Sahai (GS) proof system [18] comes in handy when proving satisfiability of quadratic equations over bilinear groups. Moreover, GS proofs are randomizable and can be adapted to satisfy some modifications of the public equations. While it provides security in the CRS model, it is easy to make it public coin by deriving the CRS from a random oracle.

In this paper, we only rely on the satisfiability of *linear* pairing-product equations (PPE) and *linear* multi-scalar equations in $\mathbb{G}$ (MSE). That is, we would like to commit to groups elements $X_1, \ldots, X_m \in \mathbb{G}$ or to scalars $y_1, \ldots, \hat{y}_n \in \mathbb{Z}_p$ and prove that they satisfy respectively one of the following equations

$$\prod_{i=1}^{m} e(X_i, \hat{B}_i) = T \qquad\qquad \prod_{i=1}^{n} A_i^{y_i} = U \tag{5}$$

for some public elements $\hat{B}_1, \ldots, \hat{B}_m \in \hat{\mathbb{G}}$, $A_1, \ldots, A_n, U \in \mathbb{G}$, and $T \in \mathbb{G}_T$. For instance, the verification equation (4) of the LHSP signature above is a PPE. In our TREnc, we will commit to elements like $Z, R \in \mathbb{G}$. Note that LHSP equation is linear and $T = \prod_{i=1}^n e(M_i, \hat{g}_i)$ above.

For our purpose, we only recall some of the Groth-Sahai algorithms, and let the others implicit like the verification.

Gen(pp): given the public parameter pp, choose $\boldsymbol{u}_1 = (u_{1,1}, u_{1,2})$, $\boldsymbol{u}_2 = (u_{2,1}, u_{2,2}) \in \mathbb{G}^2$ and $\boldsymbol{\varphi} = (\varphi_1, \varphi_2)$, $\boldsymbol{\psi} = (\psi_1, \psi_2) \in \hat{\mathbb{G}}^2$. The crs of the proof system is given by $\mathbf{crs}_{\mathsf{PPE}} = (\boldsymbol{u}_1, \boldsymbol{u}_2)$ and $\mathbf{crs}_{\mathsf{MSE}} = (\boldsymbol{\varphi}, \boldsymbol{\psi})$.

Com(crs$_{\mathsf{PPE}}$, $\boldsymbol{X}$; $\boldsymbol{r}$): to commit to a group element X in $\mathbb{G}$ with randomness $r = (r_1, r_2) \in \mathbb{Z}_p$, compute $\boldsymbol{C} = \iota(X)\boldsymbol{u}_1^{r_1}\boldsymbol{u}_2^{r_2}$. Let $\mathsf{aux} = r$.

com(crs$_{\mathsf{MS}}$, $\boldsymbol{y}$; $\boldsymbol{\rho}$): to commit to a scalar y in $\mathbb{Z}_p$ with randomness $\rho \in \mathbb{Z}_p$, compute $\hat{\boldsymbol{c}} = \boldsymbol{\varphi}^y\boldsymbol{\psi}^\rho$. Let $\mathsf{aux} = \rho$.

Prove(crs$_{\mathsf{PPE}}$, $(\hat{\boldsymbol{B}}_{\boldsymbol{i}})_{i=1}^{m}$, aux): Parse aux as $(r_{i1}, r_{i2})_{i=1}^m$, and computes the proof $\hat{\boldsymbol{\pi}} = (\hat{\pi}_1, \hat{\pi}_2)$ in $\hat{\mathbb{G}}^2$ such that $\hat{\pi}_1 = \prod_{i=1}^m \hat{B}_i^{r_{i1}}$ and $\hat{\pi}_2 = \prod_{i=1}^m \hat{B}_i^{r_{i2}}$.

prove(crs$_{\mathsf{MSE}}$, $(\boldsymbol{A}_{\boldsymbol{i}})_{i=1}^{n}$, aux): Parse aux as $(\rho_i)_{i=1}^n$, and computes the proof π in $\mathbb{G}$ such that $\pi = \prod_{i=1}^n A_i^{\rho_i}$.

Verify($\mathsf{crs}_{\mathsf{PPE}}, (\boldsymbol{C}_i)_{i=1}^m, (\hat{\boldsymbol{B}}_i)_{i=1}^m, \boldsymbol{T}, \hat{\boldsymbol{\pi}}$): Parse the proof $\hat{\boldsymbol{\pi}}$ as $(\hat{\pi}_1, \hat{\pi}_2)$ and the CRS as $\mathsf{crs}_{\mathsf{PPE}} = (\boldsymbol{u}_1, \boldsymbol{u}_2)$, and check

$$\prod_{i=1}^{m} E_1(\boldsymbol{C}_i, \hat{B}_i) = T \cdot E_1(\boldsymbol{u}_1, \pi_1) \cdot E_1(\boldsymbol{u}_2, \pi_2).$$

verify($\mathsf{crs}_{\mathsf{MSE}}, (\hat{c}_i)_{i=1}^n, (\boldsymbol{A}_i)_{i=1}^n, \boldsymbol{U}, \boldsymbol{\pi}$): Parse the CRS as $\mathsf{crs}_{\mathsf{MSE}} = (\boldsymbol{\varphi}, \boldsymbol{\psi})$, and check

$$\prod_{i=1}^{n} E_2(A_i, \hat{\boldsymbol{c}}_i) = E_2(U, \boldsymbol{\varphi}) \cdot E_2(\pi, \boldsymbol{\psi}).$$

In the *hiding mode*, $(\boldsymbol{u}_1, \boldsymbol{u}_2)$ of the CRS are linearly independent over $\mathbb{G}^2$, while (φ_1, φ_2) are linearly dependent. In that case, it is easy to see that the commitments are distributed as perfectly hiding commitments. Moreover, the Groth-Sahai proofs are perfectly witness indistinguishable (WI): proofs can be explained for any $X_1, \ldots, X_m \in \mathbb{G}$ or $y_1, \ldots, y_n \in \mathbb{Z}_p$ satisfying the PPE or the MSE, respectively, with equiprobability.

In the *extractable mode*, $(\boldsymbol{u}_1, \boldsymbol{u}_2)$ of the CRS are linearly dependent over $\mathbb{G}^2$, while (φ_1, φ_2) are linearly independent. In that case, it is easy to see that the commitments are distributed as ElGamal ciphertexts. Moreover, the Groth-Sahai proofs are perfectly sound: extracted $X_1, \ldots, X_m \in \mathbb{G}$ satisfy the PPE and the exponents of the extracted $\hat{f}^{y_1}, \ldots, \hat{f}^{y_n} \in \hat{\mathbb{G}}$, for some $\hat{f} \in \hat{\mathbb{G}}$, satisfy the MSE, both with probability 1. Both modes are indistinguishable under the SXDH assumption.

Furthermore, it is well known that Groth-Sahai (commitments and) proofs are perfectly re-randomizable, and malleable. For a single linear equation, linear combination of proofs is a proof for the same linear combination of the witness groups elements as the verification equations are homomorphic. Other modifications are possible which modify the equations of the statement and adapt the commitments and the proofs [11,16]. In this paper, we will also adapt the CRS in a verifiable way.

3 Construction

In this section, we describe our first construction supporting the encryption of ℓ bits per ciphertext. Due to space limitations, we defer the description of our construction supporting the efficient encryption of group elements to Appendix 4. Since the constructions in the standard model and in the random oracle are very close, we provide a single description and highlight where the schemes diverge.

3.1 Intuition

We provide a thorough intuition about our first TREnc construction in the random oracle to encrypt ℓ-bit strings.

A ciphertext is of the form $\mathsf{CT} = (\mathbf{c}, \mathbf{ovk}, \boldsymbol{\sigma}_{\mathsf{trace}}, \boldsymbol{\pi}_{\mathsf{valid}})$, where $\mathbf{c}$ is a chosen-plaintext (CPA) secure encryption of the message, $\mathbf{ovk}$ is a one-time LHSP verification key that is freshly generated at the encryption time, $\boldsymbol{\sigma}_{\mathsf{trace}}$ is the component offering traceability, and $\boldsymbol{\pi}_{\mathsf{valid}}$ is a tag-based randomizable proof with associated tag $\mathbf{ovk}$ that the CPA part $\mathbf{c}$ is well-formed. Below, we describe the structure of these four components in more details and provide the intuition about the security of the scheme. Eventually, we offer a high level outline of how winning the TCCA experiment tightly reduces to solving the SXDH problem in the random oracle model.

To encrypt ℓ-bit messages $\boldsymbol{m} = (m_1, \ldots, m_\ell)$, we start by computing an homomorphic ElGamal-like encryption with randomness-reuse as $\mathbf{c} = (d_1, d_2, c_1, \ldots, c_\ell) \in \mathbb{G}^{\ell+2}$ such that

$$\mathbf{c} = (1, 1, g_1^{m_1}, \ldots, g_\ell^{m_\ell}) \cdot (g, h, f_1, \ldots, f_\ell)^\theta,$$

where $f_i = g^{\alpha_i} h^{\beta_i}$ and g_i are part of the public key, for all $1 \leq i \leq \ell$, and the secret key is $\mathsf{SK} = (\mathsf{sk}_i)_{i=1}^\ell = (\alpha_1, \beta_1, \ldots, \alpha_\ell, \beta_\ell)$ which allows decrypting $\mathbf{c}$ to $g_1^{m_1}, \ldots, g_\ell^{m_\ell}$. Note that usual ℓ ElGamal encryptions would require ℓ independent random scalars θ_i and would lead to 2ℓ elements of $\mathbb{G}$ instead of $\ell+2$ with a single randomness θ. The use of two elements $(d_1, d_2) = (g, h)^\theta$ allows us to show the traceability property as explained later.

The tracing part $\boldsymbol{\sigma}_{\mathsf{trace}}$ prevents the malleability of the message-carrying components $g_1^{m_1}, \ldots, g_\ell^{m_\ell}$ while keeping the randomizability of the ciphertext $\mathbf{c}$. It relies on the LHSP signature scheme recalled in Section 2.3 to authenticate the following vectors: the vector $(g, c_1, \ldots, c_\ell, d_1)$ from the CPA encryption, and the vector $(1, f_1, \ldots, f_\ell, g)$ for randomizing the CPA part. The one-time flavor of the LHSP signatures are enough since the verification key **ovk** is only used once to authenticate the linear span generated by these vectors. Therefore, **ovk** is computed in each encryption from a new uniform one-time secret key **osk** defined as the link key lk of the TREnc ciphertext, and **ovk** is the corresponding trace. While one can derive a valid signature on any linear combination of these vectors, we will reject those for which the first component differs from g. This has the effect of forcing the exponent coefficient of $(g, c_1, \ldots, c_\ell, d)$ to 1, meaning that the vector of the underlying message-carrying part $(g, g_1^{m_1}, \ldots, g_\ell^{m_\ell}, 1)$ remains unchanged as desired. We note that is unnecessary to include d_2 and h in the above two authenticated vectors because the validity proof $\boldsymbol{\pi}_{\mathsf{valid}}$ that we will discuss later already ensures that $(d_1, d_2) = (g, h)^\theta$. Therefore, any randomization of d_1 is enforced to be carried on d_2, and we do not need to further imply that a second time by the unforgeability of the LHSP signatures. To reduce the linear size of $\boldsymbol{\sigma}_{\mathsf{trace}}$ in ℓ to a constant, we shrink these vectors in a way to keep the traceability property brought by the LHSP signatures. Since each $c_i = g_i^{m_i} f_i^\theta$ relies on an independent basis g_i, we can shrink these components into a single group element $c = \prod_{i=1}^{\ell} c_i$ and combine the one-time LHSP unforgeability with the hardness of computing discrete logarithms. Following this idea, we get the shrunk vectors (g, c, d_1) and $(1, f, g)$, where $f = \prod_{i=1}^{\ell} f_i$, that still allows randomizing $\mathbf{c}$ and adapting its LHSP signature accordingly. Since $c = g_1^{m_1} \cdots g_\ell^{m_\ell} f^\theta$ can be seen as a binding commitment of the message (even given $d_1 = g^\theta$ and $d_2 = h^\theta$) and the signature technique fixes $M = g_1^{m_1} \cdots g_\ell^{m_\ell}$, any (valid) ciphertext sharing the same trace **ovk** must decrypt to $\mathbf{m}$, except if we get another discrete-log representation M.

Although the above strategy looks enough to ensure traceability, we also need to be able to authenticate an apparently randomized ciphertext for an adversarially chosen trace **ovk** to ensure the TCCA notion. That is, in the TCCA proof we still have to produce a valid-looking randomization of CT with trace **ovk** while we have to inject an SXDH instance during the simulated randomization of $\mathbf{c}$. One attempt to facilitate this fake randomization is to rely on a zero-knowledge proof of knowledge of **osk**. Given an extracted **osk**, it is easy to authenticate any vectors even those that are not a proper randomization of $\mathbf{c}$, and no one would notice that if the modified $\mathbf{c}$ is indistinguishable of $\mathbf{c}$. However, such proofs are costly and even if our security analysis relies on the random oracle model, we still want to rely on online extraction only without rewinding technique and practical efficiency. Therefore, we follow [14] and add a third authenticated vector in $\boldsymbol{\sigma}_{\mathsf{trace}}$ that can only be used in the reduction, without compromising traceability. This third vector is $(1, F, G)$ for random group elements $F, G \in \mathbb{G}$, but unlike [14] they are derived from the trace for each ciphertext from the random oracle. This allows us to have a tight reduction to SXDH from the TCCA game as explained later since we can recycle the same argument with fresh (F, G) as long as their distributions can be correlated. In part, this is due to the fact that we build $\boldsymbol{\pi}_{\mathsf{valid}}$ as a tight unbounded simulation-sound tag-based argument, where the trace **ovk** also plays the role of the tag. Back to traceability, it is infeasible to use $(1, F, G)$ in order to randomize the CPA part without computing a valid proof $\boldsymbol{\pi}_{\mathsf{valid}}$ that the randomized $\mathbf{c}$ is still honestly computed. Nevertheless, in the TCCA proof, computing proofs of false statements is feasible, and this is in $(1, F, G)$ that we can inject a related SXDH instance from a single instance thanks to the random-self reducibility property. Now, we are able to explain why $\mathsf{sk}_i = (\alpha_i, \beta_i)$ and $\mathbf{c}$ contains a component d_2. If $\mathsf{sk}_i = \alpha_i$ with $f_i = g^{\alpha_i}$ and $\mathbf{c} = (d_1, (c_i)_{i=1}^{\ell})$ one could not show traceability. To show traceability, we at least have to ensure that $M = g_1^{m_1} \cdots g_\ell^{m_\ell}$ in $c = \prod_{i=1}^{\ell} c_i$ cannot be modified. Even if $\boldsymbol{\pi}_{\mathsf{valid}}$ must imply the validity of the apparently randomized $\mathbf{c}$, that does not mean that it cannot be of the form $(d_1', (c_i')_{i=}^{\ell}) = (g^{\theta+\theta'}, (g_i^{m_i'} f_i^{\theta+\theta'})_{i=1}^{\ell}$ with $M \neq M' = g_1^{m_1'} \cdots g_\ell^{m_\ell'}$. To show that this reduces to forging an LHSP signature, we first have to make sure that (g, c', d_1') with $c' = \prod_{i=1}^{\ell} c_i'$ is outside the linear vector space spanned by the signed three vectors. However, if $(1, F, G)$ is random it is not in the linear vector space spanned by (g, c, d_1) and $(1, f, g)$ with overwhelming probability. That means that by definition, there is no forgery as all the vector space $\mathbb{G}^3$ is already authenticated. We thus have to turn back $(1, F, G)$ in the two-dimensional vector space generated by (g, c, d_1) and $(1, f, g)$ in the proof, which leaves us with a single possibility of $(1, F, G) = (1, f, g)^\mu$ for any (random) $\mu \in \mathbb{Z}_q$. Now, it is easy to see why traceability cannot hold since we would have $F = G^\alpha$ with $\alpha = \sum_{i=1}^{\ell} \alpha_i$ and the adversary is given $\mathsf{SK} = (\mathsf{sk}_i)_{i=1}^{\ell}$. Our additional elements $h \in \mathsf{PK}$ and thus $d_2 \in \mathbf{c}$ ensure that no party knows neither the discrete logarithm of f_i in base g nor the one of F in base G.

Before completely moving to $\boldsymbol{\pi}_{\mathsf{valid}}$, we still have to bring a last modification to $\boldsymbol{\sigma}_{\mathsf{trace}}$ to achieve the TCCA security. Since the reason and the way to circumvent the last issue is like [14], we simply go quickly through it. The LHSP signature on the first vector (g, c, d) derived from $\mathbf{c}$ must only be available through a witness indistinguishable proof of knowledge (that must also be randomizable). That is because given two valid ciphertexts for the same trace **ovk** in a challenge phase, we are not able to check whether the adversary managed to produce two first LHSP signatures on their respective first vector without introducing a subliminal information into them. Since adapting these signatures to the randomization of the CPA part would not remove this information, the adversary could be able to distinguish which ciphertext has been processed. Therefore, $\boldsymbol{\sigma}_{\mathsf{trace}}$ contains a Groth-Sahai proof of the LHSP signature on (g, c, d) instead of the signature itself. It is well-known that the Groth-Sahai proof system offers all these properties for group elements. Indeed, if the CRS available in the public key, and denoted $\mathbf{crs}_\sigma$, is in the hiding mode, randomized proofs are perfectly redistributed among all valid proofs of the same statement. Moreover, $\mathbf{crs}_\sigma$ can safely be derived from the random oracle.

Now, we turn to the randomizable proof $\boldsymbol{\pi}_{\mathsf{valid}}$ that $\boldsymbol{c}$ is well-formed. To ensure that each triple (d_1, d_2, c_i) for $i \in [\ell]$ is indeed an ElGamal-like encryption of a bit m_i we use the randomizable Groth-Sahai system for scalars with another CRS $\mathbf{crs}_\pi$. This CRS can also be derived from the random oracle to ensure no one knows any hidden discrete-log relation. We observe that the each honest triple satisfies

$$d_1 = g^\theta \qquad d_2 = h^\theta \qquad c_i = g_i^{m_i} f_i^\theta \tag{6}$$

$$= d_1^{m_i} g^{\vartheta_i} \qquad\qquad = c_i^{m_i} f_i^{\vartheta_i}, \tag{7}$$

where $\vartheta_i = \theta(1 - m_i)$. It turns out that satisfying these equations implies that (d_1, d_2, c_i) is indeed an encryption of a bit. Intuitively, Equation 6 implies that the ciphertext (d_1, d_2, c_i) encrypts some message $m_i \in \mathbb{Z}_q$, and Equation 7 then implies that $(d_1, c_i)^{(1-m_i)}$ is an ElGamal encryption of 0. This leads to $m_i(1 - m_i) = 0$ and thus $m_i \in \{0, 1\}$ since $\mathbb{Z}_q$ is a field. This overall implication essentially holds because there is no g_i component in the equation $c_i = c_i^{m_i} f_i^{\vartheta_i}$ and there is no need to prove that $\vartheta_i = \theta(1 - m_i)$; only the existence of some common ϑ_i matters. For such kinds of (linear) statements about scalars, the Groth-Sahai proofs are zero-knowledge. This is fundamental as witness indistinguishability only would not make our TCCA security analysis going through. That is because a fake randomization of $\mathbf{c}$ with F and G will turn it outside the set of true statements for which there exist no witness to prove validity. (Besides, true statements are in one-to-one correspondence with their witnesses.) Moreover, the GS proof can be adapted when $\mathbf{c}$ is faithfully randomized. However, zero-knowledge is not enough in our case as we also need the proofs to be simulation-sound. More precisely, our tag-based proofs are even simulation-sound extractable in $\mathbf{m}$, which means that even if we prove some false statement related to a tag **ovk**, the adversary remains unable to produce a valid proof related to another tag and for which we cannot extract a witness $\mathbf{m}$ (which implies that it cannot produce a valid proof of a false statement). Even more, this should hold without being able to program the tag in a specific way since it is chosen by the adversary. That means techniques such as programmable hash functions would not help not only because we target a tight reduction, but because we must be able to decrypt without SK at some point in the transitions of games even for a tag **ovk** that can be used *later* in a challenge query. Therefore, it is useless to rely on (tight) techniques that try to partition the distribution of tags for which simulation is possible from those that lead to extractability. We need both together, and in a tight fashion for all the challenge queries.

We rely on a well-known technique turning a ZK proof into a simulation-sound proof. To compile the ZK proof, we make on OR-proof that either the statement hold or that "I know on opening of some public commitment" related to a tag derived from a part of the ciphertext. The idea behind introducing a tag is to prevent the adversary from reusing a proof for another tag than the one derived from the ciphertext. In our case, the tag is defined as the trace of the ciphertext since **ovk** cannot be reused through different ciphertexts that are not a randomization of each another. Conversely, we also need to adapt and randomize the validity proof when we randomize a ciphertext (that then keep the same trace). Now, we turn to explaining the structure of our validity proof step by step.

First, we define the tag of the proof as the trace of the ciphertext, and from the tag we derive a commitment C as a single random group element seen as a Pedersen commitment. That way, an opening of C can only be associated to the randomization of ciphertexts. More technically, this strategy allows separating simulated proofs for a set of tags from adversarially generated valid proofs for other tags that can appear in follow-up decryption queries. However, in the TCCA game, the adversary may query decryption of ciphertexts with a trace that will be present in a later challenge query. This is where we rely on the perfectly hiding property of the Pedersen commitments: as long as we do not provide any information about an opening of a commitment C related to a trace $\mathsf{\textbf{ovk}}$, the opening that we can program thanks to the random oracle remains statistically unpredictable. Therefore, even if we have to produce an opening of C to simulate the randomization of CT_b when emulating the answer to a later challenge query, the adversary could not have produced a previous opening of C (so, for the same trace) without allowing us to break the binding property of the Pedersen commitment. This argument will be used in the TCCA proof of security to show that the adversary is forced to proof the branch of the OR proof related to the right structure of the CPA ciphertext part $\mathsf{\textbf{c}}$. Nevertheless, we should rely on this binding property with care since the Groth-Sahai proof of the openings does not allow extracting the witnesses when they are scalars. Fortunately, if we see the commitment as $C = g^x h^y$ for random scalars $x, y \in \mathbb{Z}_p$ (programmed by a random oracle), we observe that it is enough to extract $(X = \hat{g}^x, Y = \hat{g}^y)$ seen as a modified opening satisfying $e(C, \hat{g}) = e(g, X)e(h, Y)$. It is easy to see that two different valid openings (X_1, Y_1) and (X_2, Y_2) lead to a Double Pairing solution $1 = e(g, X_1/X_2)e(h, Y_1/Y_2)$ that breaks SXDH.

Second, in the TCCA security proof we also need a way to switch between the branches being proved during the computation of the randomized challenge ciphertexts. Indeed, while the first step above shows that the adversary cannot produce a valid proof (for a trace of a ciphertext queried for pre-challenge decryption) related to the branch including C, we have to be able to "activate" this branch at some point during the emulation of the randomization in the challenge phases. That is, by apparently randomizing the ciphertext and its tag-based proof for the underlying CPA part, we must be able to simulate the proof by proving knowledge of an opening of C (because we will later make the CPA part independent of the given $\mathsf{\textbf{c}}$). To make this transition possible and indistinguishable, we rely on the GS-based OR-proof technique due to Rafols [23] that we adapt to our switching-branch randomization. The idea behind [23] is to let the prover a degree of freedom to generate a Groth-Sahai CRS $\mathsf{\textbf{crs}}_{\mathsf{sim}}$ allowing simulating proofs (here for C) but forcing the complement CRS $\mathsf{\textbf{crs}}_{\mathsf{sound}}$ to be in the perfect soundness mode. These CRSes are related by a simple relation like $\mathsf{\textbf{crs}}_\pi = \mathsf{\textbf{crs}}_{\mathsf{sound}}\mathsf{\textbf{crs}}_{\mathsf{sim}}$, where $\mathsf{\textbf{crs}}_\pi$ is the (fixed) CRS given in the public key. As long as $\mathsf{\textbf{crs}}_\pi$ is the binding mode, at most a single CRS between $\mathsf{\textbf{crs}}_{\mathsf{sound}}$ and $\mathsf{\textbf{crs}}_{\mathsf{sim}}$ can be in the hiding mode, and for which simulating is possible. An honest prover would thus use the witness to prove the statement for each (d_1, d_2, c_i) with $\mathsf{\textbf{crs}}_{\mathsf{sound}}$ and simulate the proof for C with $\mathsf{\textbf{crs}}_{\mathsf{sim}}$. This technique hides which branch of the statement is really proven and which is simulated. To make this technique compatible with our needs, not only the Groth-Sahai proofs computed from $\mathsf{\textbf{crs}}_{\mathsf{sound}}$ and $\mathsf{\textbf{crs}}_{\mathsf{sim}}$ have to be randomizable but these CRSes as well. Fortunately, GS proofs support such kinds of malleability and we can randomize $\mathsf{\textbf{crs}}_{\mathsf{sound}}$ and $\mathsf{\textbf{crs}}_{\mathsf{sim}}$ and adapt the proofs related to them before randomizing the adapted proofs related to this refresh. Although it is almost straightforward to verify this property, we encounter another technical obstacle preventing us from indistinguishably switching the mode of the CRSes $\mathsf{\textbf{crs}}_{\mathsf{sound}}$ and $\mathsf{\textbf{crs}}_{\mathsf{sim}}$ given in the challenge while this is exactly what we should do to allow switching the branches being proved. To tackle this problem, we will modify the statement of the branch containing C for simulation.

To simplify notations, let $\mathsf{\textbf{crs}}_0$ be the CRS involved in showing the validity of the CPA ciphertext $\mathsf{\textbf{c}}$ and $\mathsf{\textbf{crs}}_1$ be the CRS involved in proving knowledge of an opening of the commitment C and which should be simulated. The general simulator of [23] first turns $\mathsf{\textbf{crs}}_\pi$ included the public key in the hiding mode so that it is feasible to have both CRSes $\mathsf{\textbf{crs}}_0$ ans $\mathsf{\textbf{crs}}_1$ in the hiding mode and satisfying $\mathsf{\textbf{crs}}_\pi = \mathsf{\textbf{crs}}_0\mathsf{\textbf{crs}}_1$. This change allows simulating both branches of the OR proof: the validity of $\mathsf{\textbf{c}}$ and the opening of C. All these steps can be applied to our case but the problem to tackle comes next. While hiding-mode CRSes $\mathsf{\textbf{crs}}_0$ and $\mathsf{\textbf{crs}}_1$ can be regenerated from scratch without randomizing them, nothing ensures that those given in the ciphertexts by the adversary in a challenge phase are not both in the binding modes. If they are in the binding mode, we are stuck in the TCCA security proof since we cannot show the transition be indistinguishable. Assuming that the adversarial crs_1 is binding, it is tempting to think that everything works since we could extract another opening of C. However, this needs a trapdoor related to crs_π to extract partial witness. But, at that point in the security proof, we will still need to turn it back in the binding mode later. That is because, we want to reach a point where we use the witness for C and simulate the proof for $\mathbf{c}$ but so that no adversarially generated proofs for $\mathbf{c}$ is related to a crs_0 in the hiding mode allowing proving a false statement. If we were to use the trapdoor, we could no more prove the next transition for crs_π. We thus need a technique to force the components crs_0 and crs_1 of the challenge ciphertexts to be both in the hiding mode when we would like to modify the distribution of crs_π back and forth from the binding mode to the hiding mode. Our simple idea is to add a statement in the simulated branch of the OR-proof (that already includes the commitment C) for which we control whether it is true or false. Controlling the veracity of the statement is the key ingredient because of the following. Before our first transition for crs_π we can program this statement to be false so that the validity of $\boldsymbol{\pi}_{\mathsf{valid}}$ always imply that crs_1 is in the hiding mode since otherwise the proof would be perfectly sound, leading to a contradiction. When we will program crs_π in the hiding mode, so will be all the crs_0 and crs_1 computed by the adversary, and our simulated proof can be indistinguishably introduced in the game hops as explained above. Before turning back crs_π in the binding mode and keep simulating the ciphertext branch, we simply have to program the additional statement to be true and to use a related witness. Obviously, this strategy works as long as flipping from the statement from true to false, and conversely, is indistinguishable. Still, this is easy to achieve: it suffices to generate an additional pair of random elements (A, B) from the random oracle in the public key and define the additional statement as "(A, B) is an additive ElGamal encryption of zero" and everything works under the SXDH assumption. It is worth mentioning that C could not play this role since there always exists an opening for C, making the statement for C always true.

On achieving tightness. The compression of the vectors composing the matrix T makes the task of achieving tight TCCA security harder. Indeed, if the matrix was with $\ell + 1$ rows in addition to the first row carrying the vector from the CPA part, having a sub-square matrix of full rank will allow us to directly redistribute all the CPA parts individually for each index $i \in [\ell]$ by "over-randomization" with all the rows with additional random coins θ'_i, for each $i \in [\ell]$. Moreover, with our matrices of dimension 3 already gives a linear combination of potential random group elements that can be used to make the randomized CPA part independent of the challenger's choice for all the indexes $i \in [\ell]$ and simultaneously for all the q challenge queries to avoid a tightness loss proportional to q. Fortunately, we are able to use the random-self reducibility of SXDH to create additional random group elements to make all the c'_i independent.

3.2 Description

Let Setup be the bilinear group public parameter generation algorithm of Section 2.2. Let also poly' be a polynomial and Hash a collision-resistant hash function. We simultaneously define a construction in the standard model and its variant in the ROM where Hash is modeled as a random oracle.

$\mathsf{Gen}(1^\lambda)$: Run $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ and let $\ell = \mathsf{poly}'(\lambda)$.

1. Generate $A, B, C_0, C_1, F, G, (g_i)_{i=1}^{\ell}, h \leftarrow\!\!\$\ \mathbb{G}^{\ell+7}$ and $\hat{h} \leftarrow\!\!\$\ \hat{\mathbb{G}}$. *In the ROM construction*, ignore C_0, C_1, F, G and generate all the other values from $\mathsf{Hash}(g, \hat{g}, \ell)$ instead, where $g, \hat{g} \in \mathsf{pp}$.
2. Pick a random pair $\mathsf{sk}_i = (\alpha_i, \beta_i) \leftarrow\!\!\$\ \mathbb{Z}_p^2$ and compute $f_i = g^{\alpha_i} h^{\beta_i} \in \mathbb{G}$, for each $i \in [\ell]$.
3. Generate two tuples of 4 random group elements $(\mathsf{crs}_\sigma, \mathsf{crs}_\pi) \leftarrow\!\!\$\ \mathbb{G}^4 \times \hat{\mathbb{G}}^4$ such that $\mathsf{crs}_\sigma = (\boldsymbol{u}_1, \boldsymbol{u}_2)$ is seen as a Groth-Sahai CRS to commit to group elements over $\mathbb{G}$ and $\mathsf{crs}_\pi = (\boldsymbol{\varphi}, \boldsymbol{\psi})$ is seen as a Groth-Sahai CRS to commit to scalars over $\hat{\mathbb{G}}$. *In the ROM construction*, generate these CRSes from $\mathsf{Hash}(g, \hat{g}, (f_i)_{i=1}^{\ell})$ instead.
   For simplicity, we write Com for the commitment algorithm $\mathsf{Com}(\mathsf{crs}_\sigma, \cdot)$.

The private key consists of $\mathsf{SK} = (\mathsf{sk}_i)_{i=1}^{\ell} = (\alpha_i, \beta_i)_{i=1}^{\ell}$ and the public key $\mathsf{PK} \in \mathbb{G}^{12+2\ell} \times \hat{\mathbb{G}}^6$ is

$$\mathsf{PK} = \Big(g, h, (g_i, f_i)_{i=1}^{\ell}, A, B, C_0, C_1, F, G, \mathbf{crs}_\sigma, \mathbf{crs}_\pi, \hat{g}, \hat{h}\Big) .$$

In the ROM construction, the public key can be derived from pp and $(f_i)_{i=1}^{\ell} \in \mathbb{G}^{\ell}$ only and there is no value C_0, C_1, F, G in PK.

Enc(PK, $\boldsymbol{m}$): Given a message $\boldsymbol{m} = (m_1, \ldots, m_\ell) \in \{0,1\}^\ell$ to encrypt, first run $\underline{\mathsf{LGen}(\mathsf{PK})}$: Generate a key pair $(\mathbf{osk}, \mathbf{ovk})$ for the one-time LHSP recalled in Section 2.3 from the public generators $\hat{g}, \hat{h}$ in order to sign vectors of dimension 3. Let $\mathsf{lk} = \mathbf{osk} = \{(\eta_i, \zeta_i)\}_{i=1}^{3}$ be the private key, of which the corresponding public key is $\mathbf{ovk} = \{\hat{l}_i\}_{i=1}^{3}$. From $\mathbf{ovk}$, compute $\tau = \mathsf{Hash}(\hat{g}, \mathbf{ovk})$ and $C = C_0^{\tau} C_1$. *In the ROM construction*, derive $(C, F, G) = \mathsf{Hash}(\hat{g}, \mathbf{ovk})$.

Then, conduct the following steps of $\underline{\mathsf{LEnc}(\mathsf{PK}, \mathsf{lk}, \boldsymbol{m})}$:

1. Pick $\theta \leftarrow\!\$\, \mathbb{Z}_p$ and compute the CPA encryption $\mathbf{c} = (d_1, d_2, (c_i)_{i=1}^{\ell})$, where $d_1 = g^\theta$, $d_2 = h^\theta$ and $c_i = g_i^{m_i} f_i^{\theta}$ for each $i \in [\ell]$. Keep the random coin θ.
   Next steps 2-3 are dedicated to the tracing part $\boldsymbol{\sigma}_{\mathsf{trace}}$.
2. To allow tracing, use $\mathsf{lk} = \mathbf{osk}$ to authenticate the row space of the matrix $\mathbf{T} = \big(T_{i,j}\big)_{1 \le i,j \le 3}$
$$\mathbf{T} = \begin{pmatrix} g & c & d_1 \\ 1 & f & g \\ 1 & F & G \end{pmatrix} , \tag{8}$$
   where $c = \prod_{i=1}^{\ell} c_i$ and $f = \prod_{i=1}^{\ell} f_i$. Namely, sign each row $\boldsymbol{T}_i = (T_{i,1}, T_{i,2}, T_{i,3})$ resulting in $\boldsymbol{\sigma} = (\boldsymbol{\sigma}_i)_{i=1}^{3} \in \mathbb{G}^6$, where $\boldsymbol{\sigma}_i = (R_i, S_i) \in \mathbb{G}^2$.
3. Commit to $\boldsymbol{\sigma}_1 = (R_1, S_1)$ using $\mathbf{crs}_\sigma$ as $\boldsymbol{C}_R = \mathsf{Com}(R_1)$, $\boldsymbol{C}_S = \mathsf{Com}(S_1)$. To ensure that $\boldsymbol{\sigma}_1$ is a valid one-time LHSP signature on (g, c, d_1), compute the Groth-Sahai proof $\hat{\boldsymbol{\pi}}_{sig} \in \hat{\mathbb{G}}^2$ that
$$e(R_1, \hat{g}) \cdot e(S_1, \hat{h}) = e(g, \hat{l}_1) \cdot e(c, \hat{l}_2) \cdot e(d_1, \hat{l}_3) .$$
   Next steps are dedicated to the validity proof $\boldsymbol{\pi}_{\mathsf{valid}}$.
4. For the OR-proof, generate two correlated Groth-Sahai CRSes $\mathbf{crs}_i = (\boldsymbol{\varphi}_i, \boldsymbol{\psi}) \in \hat{\mathbb{G}}^4$ to later produce the proof $\boldsymbol{\pi}_i$, for $i \in \{0,1\}$: Initialize a simulation bit b_{sim} to 1 and pick a random $\gamma \in \mathbb{Z}_q$. Then, compute $\boldsymbol{\varphi}_0$ and $\boldsymbol{\varphi}_1$ such that $\boldsymbol{\varphi} = \boldsymbol{\varphi}_0 \boldsymbol{\varphi}_1$ and $\boldsymbol{\varphi}_{b_{\mathsf{sim}}} = \boldsymbol{\psi}^{\gamma}$ over $\hat{\mathbb{G}}^2$.
5. To compute the proof $\boldsymbol{\pi}_0 = (\boldsymbol{\pi}_{\mathsf{ciph}}, \hat{\boldsymbol{c}}_{\mathsf{ciph}})$, commit to the $2\ell{+}1$ (witness) scalars θ, m_i and $\vartheta_i = \theta(1{-}m_i)$ with $\mathbf{crs}_0$, for all $i \in [\ell]$, resulting in $\hat{\boldsymbol{c}}_0 = \mathsf{com}_0(\theta; \rho_0)$, $\hat{\boldsymbol{c}}_i = \mathsf{com}_0(m_i; \rho_i)$ and $\hat{\boldsymbol{d}}_i = \mathsf{com}_0(\vartheta_i; \tau_i)$, for some random coin $\boldsymbol{\rho}_{\mathsf{ciph}} = (\rho_0, \{\rho_i, \tau_i\}_{i=1}^{\ell})$, where com_0 denotes $\mathsf{com}_{\mathsf{crs}_0}$. Let $\hat{\boldsymbol{c}}_{\mathsf{ciph}} = (\hat{\boldsymbol{c}}_0, \{\hat{\boldsymbol{c}}_i, \hat{\boldsymbol{d}}_i\}_{i=1}^{\ell}) \in \hat{\mathbb{G}}^{4\ell+2}$.
6. Compute the Groth-Sahai proof $\boldsymbol{\pi}_{\mathsf{ciph}} \in \mathbb{G}^{3\ell+2}$ on input $\boldsymbol{\rho}_{\mathsf{ciph}}$ to show that the equations (2) and (3) hold for all $i \in [\ell]$, resulting in $3\ell + 2$ equations. To be explicit, $\boldsymbol{\pi}_{\mathsf{ciph}} = (\pi_{10}, \pi_{20}, (\pi_{1i}, \pi_{2i}, \pi_{3i})_{i=1}^{\ell})$, where
$$\pi_{10} = g^{\rho_0} \qquad \pi_{20} = h^{\rho_0} \qquad \pi_{2i} = g_i^{\rho_i} f_i^{\rho_0} \tag{9}$$
$$\pi_{1i} = d_1^{\rho_i} g^{\tau_i} \qquad \pi_{3i} = c_i^{\rho_i} f_i^{\tau_i} . \tag{10}$$
7. To compute the proof $\boldsymbol{\pi}_1 = (\hat{\boldsymbol{c}}_{\mathsf{enc}}, \boldsymbol{\pi}_{\mathsf{enc}}, \hat{\boldsymbol{c}}_{\mathsf{com}}, \pi_{\mathsf{com}})$, on input $\mathbf{crs}_1$ and the simulation key γ in $\boldsymbol{\varphi}_1 = \boldsymbol{\psi}^{\gamma}$, simulate that $(A, B) \in \langle (g, h) \rangle$ and a knowledge of an opening of the commitment C.
   - For the encryption (A, B), compute a fake commitment $\hat{\boldsymbol{c}}_{\mathsf{enc}} = \boldsymbol{\psi}^{\rho_z}$ and the simulated proof $\boldsymbol{\pi}_{\mathsf{enc}} = (\pi_A, \pi_B) = (A, B)^{-\gamma} (g, h)^{\rho_z} \in \mathbb{G}^2$ as if $\hat{\boldsymbol{c}}_{\mathsf{enc}}$ was a commitment to z such that $(A, B) = (g, h)^z$.
   - For the commitment C, compute fake commitments $\hat{\boldsymbol{c}}_x = \boldsymbol{\psi}^{\rho_x}$ and $\hat{\boldsymbol{c}}_y = \boldsymbol{\psi}^{\rho_y}$ for random $\rho_x, \rho_y \in \mathbb{Z}_q$ as well as the simulated proof $\pi_{\mathsf{com}} = C^{-\gamma} g^{\rho_x} h^{\rho_y} \in \mathbb{G}$ as if $\hat{\boldsymbol{c}}_x$ and $\hat{\boldsymbol{c}}_y$ were commitments to an opening (x, y) of $C = g^x h^y$. Let $\hat{\boldsymbol{c}}_{\mathsf{com}} = (\hat{\boldsymbol{c}}_x, \hat{\boldsymbol{c}}_y)$.

Output the ciphertext

$$\mathsf{CT} = \Big(\mathbf{c}, \mathbf{ovk}, \boldsymbol{\sigma}_{\mathsf{trace}}, \boldsymbol{\pi}_{\mathsf{valid}}\Big),$$

where $\boldsymbol{\sigma}_{\mathsf{trace}} = (\boldsymbol{C}_R, \boldsymbol{C}_S, \boldsymbol{\sigma}_2, \boldsymbol{\sigma}_3, \hat{\boldsymbol{\pi}}_{sig}) \in \mathbb{G}^8 \times \hat{\mathbb{G}}^2$ and $\boldsymbol{\pi}_{\mathsf{valid}} = (\boldsymbol{\varphi}_0, \boldsymbol{\pi}_0, \boldsymbol{\pi}_1)$ with $\boldsymbol{\pi}_0 \in \mathbb{G}^{3\ell+2} \times \hat{\mathbb{G}}^{4\ell+2}$ and $\boldsymbol{\pi}_1 \in \mathbb{G}^3 \times \hat{\mathbb{G}}^6$ up to reordering.

Trace(PK, CT): Parse PK and CT as above, and output **ovk**.

Rand(PK, CT): Parse the ciphertext as above, and let (C, F, G) be computed from $\mathsf{Hash}(\hat{g}, \mathbf{ovk})$ and PK as above. Conduct the following steps:

1. Randomize the CPA part $\mathbf{c} = (d_1, d_2, (c_i)_{i=1}^{\ell})$: Pick $\theta' \leftarrow\!\!\$\, \mathbb{Z}_p$ and compute $\mathbf{c}' = (d_1', d_2', (c_i')_{i=1}^{\ell}) = (d_1 g^{\theta'}, d_2 h^{\theta'}, (c_i f_i^{\theta'})_{i=1}^{\ell}) = \mathbf{c} \cdot (g, h, (f_i)_{i=1}^{\ell})^{\theta'}$.
   Next steps 2-3 are dedicated to the tracing part $\boldsymbol{\sigma}'_{\mathsf{trace}}$.
2. Adapt the commitment $(\boldsymbol{C}_R, \boldsymbol{C}_S)$ to the signature $\boldsymbol{\sigma}_1$ so that it it becomes a commitment $(\tilde{\boldsymbol{C}}_R, \tilde{\boldsymbol{C}}_S)$ to the signature $\boldsymbol{\sigma}_1' = \boldsymbol{\sigma}_1 \boldsymbol{\sigma}_2^{\theta'}$. That is, parse $\boldsymbol{\sigma}_2$ as (R_2, S_2) and compute $\tilde{\boldsymbol{C}}_R = \boldsymbol{C}_R \cdot \iota(R_2^{\theta'})$ and $\tilde{\boldsymbol{C}}_S = \boldsymbol{C}_S \cdot \iota(S_2^{\theta'})$. Note: $\boldsymbol{\sigma}_1'$ is a valid signature on $(g, \prod c_i', d_1')$ for **opk**, and $\hat{\boldsymbol{\pi}}_{sig}$ remains a proof that $\boldsymbol{\sigma}_1' = (R_1', S_1')$ satisfies

$$e(R_1', \hat{g}) \cdot e(S_1', \hat{h}) = e(g, \hat{l}_1) \cdot e(c, \hat{l}_2) \cdot e(d_1, \hat{l}_3),$$

   where $(R_1', S_1') = (R_1, S_1) \cdot (R_2, S_2)^{\theta'}$.
3. Randomize the GS proof $((\tilde{\boldsymbol{C}}_R, \tilde{\boldsymbol{C}}_S), \hat{\boldsymbol{\pi}}_{sig})$ for the CRS $\mathbf{crs}_\sigma$ leading to $((\boldsymbol{C}'_R, \boldsymbol{C}'_S), \hat{\boldsymbol{\pi}}'_{sig})$. Let $\boldsymbol{\sigma}'_{\mathsf{trace}} = (\boldsymbol{C}'_R, \boldsymbol{C}'_S, \boldsymbol{\sigma}_2, \boldsymbol{\sigma}_3, \hat{\boldsymbol{\pi}}'_{sig})$[6] be the randomized tracing part.
   Next steps are dedicated to the validity proof $\boldsymbol{\pi}'_{\mathsf{valid}}$.
4. Randomize the correlated Groth-Sahai CRSes of the OR-proof: Given $\boldsymbol{\varphi}_0$, compute $\boldsymbol{\varphi}_1$ such that $\boldsymbol{\varphi} = \boldsymbol{\varphi}_0 \boldsymbol{\varphi}_1$ and let $\mathbf{crs}_i = (\boldsymbol{\varphi}_i, \boldsymbol{\psi})$, for $i \in \{0, 1\}$. Then, pick a random $\gamma' \in \mathbb{Z}_q$, and compute $\boldsymbol{\varphi}_0' = \boldsymbol{\varphi}_0 \boldsymbol{\psi}^{-\gamma'}$ as well as $\boldsymbol{\varphi}_1' = \boldsymbol{\varphi}_1 \boldsymbol{\psi}^{\gamma'}$. Let $\mathbf{crs}_i' = (\boldsymbol{\varphi}_i', \boldsymbol{\psi})$, for $i \in \{0, 1\}$.
5. Adapt the proof $\boldsymbol{\pi}_0 = (\boldsymbol{\pi}_{\mathsf{ciph}}, \hat{\boldsymbol{c}}_{\mathsf{ciph}})$ with respect to the randomized $\mathbf{c}'$ and $\mathbf{crs}_0'$ thanks to (θ', γ'):
   - Adapt the witness in $\hat{\boldsymbol{c}}_{\mathsf{ciph}} = (\hat{\boldsymbol{c}}_0, \{\hat{\boldsymbol{c}}_i, \hat{\boldsymbol{d}}_i\}_{i=1}^{\ell})$: Compute $\tilde{\boldsymbol{c}}_0 = \hat{\boldsymbol{c}}_0 \cdot \boldsymbol{\varphi}_0^{\theta'}$ and $\tilde{\boldsymbol{d}}_i = \hat{\boldsymbol{d}}_i \cdot (\boldsymbol{\varphi}_0 / \hat{\boldsymbol{c}}_i)^{\theta'}$, for $i = 1$ to ℓ. The proof part $\boldsymbol{\pi}_{\mathsf{ciph}}$ remains valid for $\mathbf{crs}_0$ and $\tilde{\boldsymbol{c}}_{\mathsf{ciph}} = (\tilde{\boldsymbol{c}}_0, \{\hat{\boldsymbol{c}}_i, \tilde{\boldsymbol{d}}_i\}_{i=1}^{\ell})$.
   - Adapt $\boldsymbol{\pi}_{\mathsf{ciph}} = (\pi_{10}, \pi_{20}, (\pi_{1i}, \pi_{2i}, \pi_{3i})_{i=1}^{\ell})$ to $\mathbf{crs}_0'$ by computing $\tilde{\boldsymbol{\pi}}_{\mathsf{ciph}} = \boldsymbol{\pi}_{\mathsf{ciph}} \cdot (d_1', d_2', (d_1', c_i', c_i')_{i=1}^{\ell})^{\gamma'}$.
6. Randomize the proof $\tilde{\boldsymbol{\pi}}_0 = (\tilde{\boldsymbol{\pi}}_{\mathsf{ciph}}, \tilde{\boldsymbol{c}}_{\mathsf{ciph}})$ for the CRS $\mathbf{crs}_0'$ leading to $\boldsymbol{\pi}_0' = (\boldsymbol{\pi}'_{\mathsf{ciph}}, \hat{\boldsymbol{c}}'_{\mathsf{ciph}})$.
7. Adapt the proof $\boldsymbol{\pi}_1 = (\hat{\boldsymbol{c}}_{\mathsf{enc}}, \boldsymbol{\pi}_{\mathsf{enc}}, \hat{\boldsymbol{c}}_{\mathsf{com}}, \pi_{\mathsf{com}})$ to the randomized $\mathbf{crs}_1'$ thanks to γ'. Namely, compute $(\tilde{\pi}_A, \tilde{\pi}_B, \tilde{\pi}_{\mathsf{com}}) = (\pi_A, \pi_B, \pi_{\mathsf{com}}) \cdot (A, B, C)^{-\gamma'}$, where $\boldsymbol{\pi}_{\mathsf{enc}} = (\pi_A, \pi_B)$. Then, randomize the proof $\tilde{\boldsymbol{\pi}}_1 = (\hat{\boldsymbol{c}}_{\mathsf{enc}}, \tilde{\boldsymbol{\pi}}_{\mathsf{enc}}, \hat{\boldsymbol{c}}_{\mathsf{com}}, \tilde{\pi}_{\mathsf{com}})$ for the CRS $\mathbf{crs}_1'$, where $\tilde{\boldsymbol{\pi}}_{\mathsf{enc}} = (\tilde{\pi}_A, \tilde{\pi}_B)$, leading to $\boldsymbol{\pi}_1' = (\hat{\boldsymbol{c}}'_{\mathsf{enc}}, \boldsymbol{\pi}'_{\mathsf{enc}}, \hat{\boldsymbol{c}}'_{\mathsf{com}}, \pi'_{\mathsf{com}})$.

Return the randomized ciphertext

$$\mathsf{CT}' = \Big(\mathbf{c}', \mathbf{ovk}, \boldsymbol{\sigma}'_{\mathsf{trace}}, \boldsymbol{\pi}'_{\mathsf{valid}} = (\boldsymbol{\varphi}_0', \boldsymbol{\pi}_0', \boldsymbol{\pi}_1')\Big).$$

Ver(PK, CT): Conduct the following checks:

1. Verify whether PK and CT parse properly. If not, output 0. Else compute (C, F, G) from $\mathsf{Hash}(\hat{g}, \mathbf{ovk})$ and PK as above.
   Next steps 2-3 verify the tracing part $\boldsymbol{\sigma}_{\mathsf{trace}}$.
2. Verify the validity of the signatures $\boldsymbol{\sigma}_2$ and $\boldsymbol{\sigma}_3$ on the 2 last rows $\{\boldsymbol{T}_i\}_{i=2}^{3}$ of the matrix $\mathbf{T}$ in (8) with respect to $\mathbf{ovk} = \{\hat{l}_i\}_{i=1}^{3}$. Namely, parse $\boldsymbol{\sigma}_2 = (R_2, S_2)$ and $\boldsymbol{\sigma}_3 = (R_3, S_3)$ and check the next equations:

$$e(R_2, \hat{g}) e(S_2, \hat{h}) = e(f, \hat{l}_2) e(g, \hat{l}_3)$$
$$e(R_3, \hat{g}) e(S_3, \hat{h}) = e(F, \hat{l}_2) e(G, \hat{l}_3)$$

[6] Since computing these LHSP signatures is deterministic in $\mathsf{lk} = \mathbf{osk}$, they cannot be randomized, and we do not need to randomize them to satisfy the strong randomizability notion.

3. Verify the validity of the proof $\hat{\boldsymbol{\pi}}_{sig}$ that committed variables in $\boldsymbol{C}_R$ and $\boldsymbol{C}_S$ consist of a signature on the first row (g, c, d) of $\mathbf{T}$ under ovk.
Next steps 4-6 verify the validity proof part $\boldsymbol{\pi}_{\mathsf{valid}}$.
4. Given $\boldsymbol{\varphi}_0$, compute $\boldsymbol{\varphi}_1$ such that $\boldsymbol{\varphi} = \boldsymbol{\varphi}_0\boldsymbol{\varphi}_1$ and let $\mathsf{crs}_i = (\boldsymbol{\varphi}_i, \boldsymbol{\psi})$, for $i \in \{0, 1\}$.
5. Verify the proof $\boldsymbol{\pi}_0 = (\boldsymbol{\pi}_{\mathsf{ciph}}, \hat{\boldsymbol{c}}_{\mathsf{ciph}})$. Namely, parse the commitment $\hat{\boldsymbol{c}}_{\mathsf{ciph}} = (\hat{\boldsymbol{c}}_0, \{\hat{\boldsymbol{c}}_i, \hat{\boldsymbol{d}}_i\}_{i=1}^{\ell})$ and the proof $\boldsymbol{\pi}_{\mathsf{ciph}} = (\pi_{10}, \pi_{20}, (\pi_{1i}, \pi_{2i}, \pi_{3i})_{i=1}^{\ell})$, and check

$$E_2(d_1, \boldsymbol{\varphi}_0)E_2(\pi_{10}, \boldsymbol{\psi}) = E_2(g, \hat{\boldsymbol{c}}_0)$$
$$E_2(d_2, \boldsymbol{\varphi}_0)E_2(\pi_{20}, \boldsymbol{\psi}) = E_2(h, \hat{\boldsymbol{c}}_0)$$
$$E_2(c_i, \boldsymbol{\varphi}_0)E_2(\pi_{2i}, \boldsymbol{\psi}) = E_2(g_i, \hat{\boldsymbol{c}}_i)E_2(f_i, \hat{\boldsymbol{c}}_0)$$

for Equation (2) as well as

$$E_2(d_1, \boldsymbol{\varphi}_0)E_2(\pi_{1i}, \boldsymbol{\psi}) = E_2(d_1, \hat{\boldsymbol{c}}_i)E_2(g, \hat{\boldsymbol{d}}_i)$$
$$E_2(c_i, \boldsymbol{\varphi}_0)E_2(\pi_{3i}, \boldsymbol{\psi}) = E_2(c_i, \hat{\boldsymbol{c}}_i)E_2(f_i, \hat{\boldsymbol{d}}_i)$$

for Equation (3).
6. Verify the proof $\boldsymbol{\pi}_1 = (\hat{\boldsymbol{c}}_{\mathsf{enc}}, \boldsymbol{\pi}_{\mathsf{enc}}, \hat{\boldsymbol{c}}_{\mathsf{com}}, \pi_{\mathsf{com}})$. Namely, parse $\boldsymbol{\pi}_{\mathsf{enc}} = (\pi_A, \pi_B)$ and $\hat{\boldsymbol{c}}_{\mathsf{com}} = (\hat{\boldsymbol{c}}_x, \hat{\boldsymbol{c}}_y)$, and check

$$E_2(A, \boldsymbol{\varphi}_1)E_2(\pi_A, \boldsymbol{\psi}) = E_2(g, \hat{\boldsymbol{c}}_{\mathsf{enc}})$$
$$E_2(B, \boldsymbol{\varphi}_1)E_2(\pi_B, \boldsymbol{\psi}) = E_2(h, \hat{\boldsymbol{c}}_{\mathsf{enc}})$$
$$E_2(C, \boldsymbol{\varphi}_1)E_2(\pi_{\mathsf{com}}, \boldsymbol{\psi}) = E_2(g, \hat{\boldsymbol{c}}_x)E_2(h, \hat{\boldsymbol{c}}_y)$$

Output 1 if all these checks pass, otherwise, output 0.

Dec(SK, PK, CT): If $\mathsf{Ver}(\mathsf{PK}, \mathsf{CT}) = 0$, output $\perp$. Otherwise, given $\mathsf{SK} = (\alpha_i, \beta_i)_{i=1}^{\ell}$ and $\mathbf{c} = (d_1, d_2, (c_i)_{i=1}^{\ell})$ included in CT, compute and output $\mathbf{m} = (m_i)_{i=1}^{\ell}$ such that $m_i = 0$ if $c_i = d_1^{\alpha_i} d_2^{\beta_i}$, and $m_i = 1$, otherwise.

The ciphertext consists of $4\ell + 15$ group elements in $\mathbb{G}$ and $4\ell + 15$ group elements in $\hat{\mathbb{G}}$, for any $\ell = \mathsf{poly}'(\lambda)$. Correctness of our TREnc follows by inspection and the correctness of the related cryptographic building blocks.

Theorem 1. *In the standard model, our TREnc is statistically verifiable and statistically strongly randomizable, and computationally traceable and TCCA secure under the SXDH assumption relative to* Setup *if* Hash *is collision-resistant. In the random oracle model, the TREnc variant is statistically verifiable and statistically strongly randomizable, and computationally traceable and almost-tightly TCCA secure in the multi-challenge setting under the SXDH assumption relative to* Setup*, where* $\varepsilon_{\mathtt{tcca}} \leq \frac{1}{2} + (\ell + 6)\varepsilon_{\mathtt{sxdh}} + \varepsilon(\lambda)$ *with negligible* ε.

*Proof (**Verifiability of Theorem 1**).* Let $\mathcal{A}$ be an adversary against the verifiability notion of Definition 2 such that, given $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{Gen}(1^\lambda)$, $\mathcal{A}$ produces a ciphertext CT. Next, we show that if CT is valid then $\mathsf{CT} \in \mathsf{Enc}(\mathsf{PK})$ but with probability $\varepsilon_{\mathtt{ver}} \leq 5/p$, where $p > 2^{\mathsf{poly}(\lambda)}$.

1. Since the distribution of $(A, B) \in \mathbb{G}^2$ in the key generation is uniform (either from the ROM or from the generation itself), we can write $A = g^z$ and $B = Dh^z$ for random $z \in \mathbb{Z}_p$ and $D \in \mathbb{G}$. Except with probability $1/p$, we have $D \neq 1$. That is the statement to prove for (A, B) is false with overwhelming probability. In that case, since GS CRS in the binding mode leads to perfectly sound proofs, the CRS $\mathsf{crs}_1 = (\boldsymbol{\varphi}_1, \boldsymbol{\psi}) \in \hat{\mathbb{G}}^4$ to commit to scalars over $\hat{\mathbb{G}}$ must be in the hiding mode. That is, $\boldsymbol{\varphi}_1 = \boldsymbol{\psi}^\gamma$, for some $\gamma \in \mathbb{Z}_p$, as in a honest run of item 4 of the encryption algorithm.
Moreover, as long as $\boldsymbol{\varphi}_1 = \boldsymbol{\psi}^\gamma$, the proof $\boldsymbol{\pi}_1 = (\hat{\boldsymbol{c}}_{\mathsf{enc}}, \boldsymbol{\pi}_{\mathsf{enc}}, \hat{\boldsymbol{c}}_{\mathsf{com}}, \pi_{\mathsf{com}})$ of item 7 can always be explained by scalars playing the role of the random coins that could have been selected during an honest run (even if nothing can be stated about their distribution). This is a fact due to the perfect witness-indistinguishability of GS proof based on a (perfectly) hiding CRS.

2. Since the distribution of $\mathsf{crs}_\pi = (\boldsymbol{\varphi}, \boldsymbol{\psi}) \in \hat{\mathbb{G}}^4$ is uniform (either from the ROM or the generation itself), we can write $\boldsymbol{\varphi} = (\hat{f}\hat{\varphi}^\xi, \hat{\varphi})$ and $\boldsymbol{\psi} = (\hat{\psi}^\xi, \hat{\psi})$ for random $\hat{f}, \hat{\varphi}, \hat{\psi} \in \hat{\mathbb{G}}$ as long as $\hat{\psi} \neq 1$, which only occurs with negligible probability $1/p$. Therefore, in item 4 of the encryption algorithm, we must have $\boldsymbol{\varphi}_0 = \boldsymbol{\varphi}\boldsymbol{\psi}^{-\gamma} = (\hat{f}\hat{\varphi}^\xi/\hat{\psi}^{\xi\gamma}, \hat{\varphi}/\hat{\psi}^\gamma)$ from the previous consideration on $\boldsymbol{\varphi}_1$. That is, ξ is an extraction trapdoor key for $\mathsf{crs}_0 = (\boldsymbol{\varphi}_0, \boldsymbol{\psi})$.
   As a consequence, by "decrypting" with ξ on top of the verification equations of item 5 of the verification algorithm we can find, if $\hat{f} \neq 1$ (which happens with probability $1/p$),
   $$\begin{aligned} e(d_1, \hat{f}) &= e(g, \hat{c}_0) & e(d_2, \hat{f}) &= e(g, \hat{c}_0) & e(c_i, \hat{f}) &= e(g_i, \hat{c}_i)e(f_i, \hat{c}_0) \\ &= e(d_1, \hat{c}_i)e(g, \hat{d}_i) & & & &= e(c_i, \hat{c}_i)e(f_i, \hat{d}_i) \end{aligned}$$
   where $\hat{c}_0, \hat{c}_i, \hat{d}_i$ are the extracted group elements from $\hat{\boldsymbol{c}}_0, \hat{\boldsymbol{c}}_i, \hat{\boldsymbol{d}}_i$ respectively. Therefore, for some $\theta, m_i, \vartheta_i \in \mathbb{Z}_p$ such that $\hat{c}_0 = \hat{f}^\theta$, $\hat{c}_i = \hat{f}^{m_i}$, $\hat{d}_i = \hat{f}^{\vartheta_i}$, we must have $d_1 = g^\theta$, $d_2 = h^\theta$, $c_i = g_i^{m_i} f_i^\theta$ from the first row, and then $g^{\theta(1-m_i)} = g^{\vartheta_i}$, $g_i^{m_i(1-m_i)} f_i^{\theta(1-m_i)} = f_i^{\vartheta_i}$ from the second, which in turn implies $\vartheta_i = \theta(1-m_i)$, and thus $m_i(1-m_i) = 0$.
   Then, the CPA part $\mathbf{c}$, crs_0, and $\boldsymbol{\pi}_0 = (\boldsymbol{\pi}_{\mathsf{ciph}}, \hat{\boldsymbol{c}}_{\mathsf{ciph}})$ can also be explained as a honest run of items 1-5-6 of the encryption algorithm for some random coins.
3. Now, we turn to ovk that should lies in the output of LGen and to the remaining items 2-3 of the encryption algorithm with the LHSP signatures and the GS proof $\boldsymbol{\sigma}_{\mathsf{trace}} = (\boldsymbol{C}_R, \boldsymbol{C}_S, \boldsymbol{\sigma}_2, \boldsymbol{\sigma}_3, \hat{\boldsymbol{\pi}}_{sig})$.
   Since crs_σ is hiding as a commitment key to commit to group element, except if it is a DH tuple, which can only occur with negligible probability $1/p$, $(\boldsymbol{C}_R, \boldsymbol{C}_S, \hat{\boldsymbol{\pi}}_{sig})$ can always be explained as an honest run for any witness satisfying the LHSP equation in (3). Now, let consider the 6 variables forming the potential $\mathsf{lk} = \mathsf{osk} = \{(\eta_i, \zeta_i)\}_{i=1}^3$ explaining ovk. This public information impose a single linear relation between η_i and ζ_i for each index. Moreover, the verification equations of item 2 of the verification algorithm impose 2 linear conditions (since the two last row of the matrix $\mathbf{T}$ in Equation 8 are linearly independent with overwhelming probability $(p-1)/p$ due to the uniform generation of $F, G \in \mathbb{G}^2$ either from the random oracle or the generation in the Gen algorithm itself. All these conditions leave one degree of freedom for either η_1 or ζ_1. Therefore, for any chosen value of η_1, we can explain ovk, and then $(\boldsymbol{C}_R, \boldsymbol{C}_S, \hat{\boldsymbol{\pi}}_{sig})$.

Consequently, the scheme is statistically verifiable. More precisely, we have $\varepsilon_{\mathtt{ver}} \leq 5/p$, with $p > 2^{\mathsf{poly}(\lambda)}$. □

*Proof (**Strong randomizability of Theorem 1**).* Our TREnc is *statistically* strongly randomizable as defined in Definition 3. The honestly generated public key is given either by $\mathsf{PK} = (g, h, (g_i, f_i)_{i=1}^\ell, A, B, C_0, C_1, F, G, \mathsf{crs}_\sigma, \mathsf{crs}_\pi, \hat{g}, \hat{h})$ generated uniformly from $\mathbb{G}^{12+2\ell} \times \hat{\mathbb{G}}^6$ or by $\mathsf{PK} = (g, h, (g_i, f_i)_{i=1}^\ell, A, B, \mathsf{crs}_\sigma, \mathsf{crs}_\pi, \hat{g}, \hat{h})$ from the ROM. Let $\boldsymbol{m} = (m_1, \ldots, m_\ell) \in \{0,1\}^\ell$ be a message and $\mathsf{lk} = \mathsf{osk} = \{(\eta_i, \zeta_i)\}_{i=1}^3 \in \mathbb{Z}_p^6$ be a link key. For any ciphertext $\mathsf{CT} \in \mathsf{LEnc}(\mathsf{PK}, \mathsf{lk}, \boldsymbol{m})$, we show that $\mathsf{Rand}(\mathsf{PK}, \mathsf{CT}) \approx_s \mathsf{LEnc}(\mathsf{PK}, \mathsf{osk}, \boldsymbol{m})$, where $\approx_s$ refers to statistical indistinguishability. Let $\varepsilon_{\mathtt{rand}}$ be the probability to distinguish the two distributions.

By assumption, the ciphertext CT can be parsed as $(\mathbf{c}, \mathsf{ovk}, \boldsymbol{\sigma}_{\mathsf{trace}}, \boldsymbol{\pi}_{\mathsf{valid}})$, where ovk, the tracing part $\boldsymbol{\sigma}_{\mathsf{trace}} = (\boldsymbol{C}_R, \boldsymbol{C}_S, \boldsymbol{\sigma}_2, \boldsymbol{\sigma}_3, \hat{\boldsymbol{\pi}}_{sig})$ and the validity proof $\boldsymbol{\pi}_{\mathsf{valid}} = (\boldsymbol{\varphi}_0, \boldsymbol{\pi}_0, \boldsymbol{\pi}_1)$ respect the computation of LEnc as described in the encryption algorithm, for some values playing the role of the random coins (irrespective of their distribution). Moreover, as in the previous proof of verifiability, crs_σ of PK is in the hiding mode to commit to group elements, and $\mathsf{crs}_\pi = (\boldsymbol{\varphi}, \boldsymbol{\psi})$ is in the binding mode to commit to scalars, except with probability $1/p$ for each case. Then, we must have:

1. The trace ovk is deterministic in $\mathsf{lk} = \mathsf{osk}$ as the verification key of the LHSP signature scheme of Section 2.3.
2. In $\boldsymbol{\sigma}_{\mathsf{trace}}$, the LHSP signatures $\boldsymbol{\sigma}_2, \boldsymbol{\sigma}_3$ on the vectors $(1, f, g)$ and $(1, F, G)$ that are the rows of the matrix $\mathbf{T}$ in Equation 2, where F and G are in the public key or are derived from $\mathsf{Hash}(\hat{g}, \mathsf{ovk})$ in the ROM variant. Those signatures satisfy the relation of the deterministic signing algorithm of Section 2.3.
3. The GS proof $(\boldsymbol{C}_R, \boldsymbol{C}_S, \hat{\boldsymbol{\pi}}_{sig})$ in $\boldsymbol{\sigma}_{\mathsf{trace}}$ is a valid proof of knowledge of a signature $\boldsymbol{\sigma}_1 = (R_1, S_1)$ on the vector in the first row of $\mathbf{T}$ composed of elements from the CPA part $\mathbf{c}$ that can be written as

$(d_1, d_2, (c_i)_{i=1}^{\ell})$, where $d_1 = g^{\theta}$, $d_2 = h^{\theta}$, and $c_i = g_i^{m_i} f_i^{\theta}$ for each $i \in [\ell]$, for some $\theta \in \mathbb{Z}_p$. This proof is perfectly witness indistinguishable and randomizable.

4. $\boldsymbol{\varphi} = \boldsymbol{\varphi}_0 \boldsymbol{\varphi}_1$ and $\boldsymbol{\varphi}_1 = \boldsymbol{\psi}^{\gamma}$ for some $\gamma \in \mathbb{Z}_p$. Therefore, $\mathbf{crs}_0 = (\boldsymbol{\varphi}_0, \boldsymbol{\psi})$ is in the binding mode, and $\mathbf{crs}_1 = (\boldsymbol{\varphi}_1, \boldsymbol{\psi})$ is in the hiding mode for scalars. Then, $\boldsymbol{\pi}_0$ is perfectly sound and $\boldsymbol{\pi}_1$ is perfectly witness indistinguishable and randomizable.

Based on these observations, we next show that the distribution of honest randomization of CT and the distribution of honest re-encryption of $\boldsymbol{m}$ with the link key lk above are identical.

On the one hand, independently of the precise values taken by θ, γ and the hidden scalars in all the GS proofs (so, even if $\theta = \gamma = 0$, for instance) the algorithm Rand perfectly randomize the CPA part $\mathbf{c}$ in item 1 as well as $\boldsymbol{\varphi}_1$ in item 4. Then, the randomized tracing part is fully redistributed among all the valid proofs with the adapted $\mathbf{c}$ (but $\boldsymbol{\sigma}_2, \boldsymbol{\sigma}_3$ are unchanged as detailed in item 3), and the CRSes $\mathbf{crs}_0$ and $\mathbf{crs}_1$ are perfectly randomized up to $\boldsymbol{\varphi} = \boldsymbol{\varphi}_0 \boldsymbol{\varphi}_1$ and remain in the same modes as in CT above. Now, we detail the distribution of the randomized $\boldsymbol{\pi}_0$ and $\boldsymbol{\pi}_1$. The witness of the proof $\boldsymbol{\pi}_0$ has been perfectly adapted to the randomized $\mathbf{c}$ in item 5 of Rand. Since the commitments $\hat{\boldsymbol{c}}_{\mathsf{ciph}}$ is perfectly randomized in the next item 6 and the linear proof part $\boldsymbol{\pi}_{\mathsf{ciph}}$ depends only on the random coins, $\boldsymbol{\pi}'_0$ is distributed as a fresh proof for the updated coins. The proof $\boldsymbol{\pi}_1$ is also fully redistributed as a fresh proof for the unmodified statement related to (A, B, C).

On the other hand, with a fresh execution of LEnc on input PK, lk and $\boldsymbol{m}$, $\boldsymbol{m}$ is unchanged, the link key lk leads to the same trace $\mathbf{ovk}$ as well as the trace-dependent value C generated as $C_0^{\mathsf{Hash}(\hat{g},\mathbf{ovk})} C_1$ or directly from the ROM, the same values F, G (either in the public key or derived from $\mathbf{ovk}$ in the ROM-based construction) and the signatures on $(1, f, g)$ and $(1, F, G)$ are uniquely determined since the LHSP signing algorithm is deterministic. That is LEnc produces the same $\boldsymbol{\sigma}_2, \boldsymbol{\sigma}_3$ as above. Moreover, the CPA part on $\boldsymbol{m}$, $\boldsymbol{\varphi}_0$, and all the GS commitments and proofs are honestly distributed by definition (with $\boldsymbol{\pi}_1$ relying on the same C). So these distributions are the same as the output of Rand as justify above when analyzing the distribution of the randomized ciphertext.

To conclude, the scheme is statistically strongly randomizable. More precisely, we have $\varepsilon_{\mathtt{rand}} \leq 2/p$. $\square$

*Proof (**Traceability of Theorem 1**).* Let $\mathcal{A}$ be an adversary against the traceability of our TREnc, and $\varepsilon_{\mathtt{trace}}$ be its winning probability in the traceability game as defined in Figure 1 (right) of Definition 4. We give a succession of hybrid games $\mathcal{H}_0, \ldots, \mathcal{H}_9$, where $\mathcal{H}_0$ is the real traceability game. For each of those games, we denote P_i the probability that $\mathcal{H}_i$ outputs 1 in the security parameter λ. Also, we define q_{ovk} as the number of distinct Hash-queries of the form $(C, F, G) \leftarrow \mathsf{Hash}(\hat{g}, \mathsf{ovk})$ made by the adversary for any triple $\mathsf{ovk} \in \hat{\mathbb{G}}^3$ in the ROM-based construction.

Game 0: This is the real traceability game where, given $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{Gen}(1^{\lambda})$, the adversary $\mathcal{A}$ chooses $\boldsymbol{m} \in \{0,1\}^{\ell}$ with $\ell = \mathsf{poly}'(\lambda)$. It gets $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{PK}, \boldsymbol{m})$ and outputs CT^*. By definition, $\mathcal{H}_0$ outputs 1 if $\mathsf{Trace}(\mathsf{PK}, \mathsf{CT}) = \mathsf{Trace}(\mathsf{PK}, \mathsf{CT}^*)$, $\mathsf{Ver}(\mathsf{PK}, \mathsf{CT}^*) = 1$, but $\mathsf{Dec}(\mathsf{SK}, C^*) \neq \boldsymbol{m}$. By definition, we have $P_0 = \varepsilon_{\mathtt{trace}}$.

Game 1: In this game, we modify $\mathcal{H}_0$ in the way we produce CT. Instead of directly computing $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{PK}, \boldsymbol{m})$, we first run $\mathsf{lk} = \mathbf{osk} = \{(\eta_i, \zeta_i)\}_{i=1}^{3} \leftarrow \mathsf{LGen}(\mathsf{PK})$ at the outset of the game and keep $\mathbf{osk}$ throughout all the game(s). Then, when the adversary requests the encryption, we compute $\mathsf{CT} \leftarrow \mathsf{LEnc}(\mathsf{PK}, \mathbf{osk}, \boldsymbol{m})$. The trace is then the public key $\mathbf{ovk} = \{\hat{l}_i\}_{i=1}^{3}$ of the LHSP signature of Section 2.3 computed from $\mathbf{osk}$. For the ROM-based case, we also assume that the challenger computes $(C, F, G) = \mathsf{Hash}(\hat{g}, \mathbf{ovk})$ at the outset of the game. By definition of TREnc, we have $P_1 = P_0$.

Game 2: In this game, we modify the generation of the Groth-Sahai CRS $\mathbf{crs}_{\sigma} = (\boldsymbol{u}_1, \boldsymbol{u}_2) \in \mathbb{G}^4$ in item 3 of the key generation by picking a random exponent $\nu \leftarrow\$\, \mathbb{Z}_p$, $\boldsymbol{u}_1 \leftarrow\$\, \mathbb{G}^2$ and computing $\boldsymbol{u}_2 = \boldsymbol{u}_1^{\nu}$. In the ROM variant, we program the random oracle to output this CRS: $(\mathbf{crs}_{\sigma}, \mathbf{crs}_{\pi}) \leftarrow \mathsf{Hash}(g, \hat{g}, (f_i)_{i=1}^{\ell})$, where the distribution of $\mathbf{crs}_{\pi}$ is unchanged. Clearly, $|P_2 - P_1| \leq \varepsilon_{\mathtt{sxdh}} + 1/p$.

Game 3: This game is as $\mathcal{H}_2$ except in the way we generate $\boldsymbol{u}_1 = (u_{1,1}, u_{1,2})$ in $\mathbf{crs}_{\sigma}$. Now, we pick a trapdoor extracting key $\tau_{\sigma} \leftarrow\$\, \mathbb{Z}_p$ and compute $u_{1,1} = u_{1,2}^{\tau_{\sigma}}$. As long as $u_{1,2} \neq 1_{\mathbb{G}}$, $u_{1,1}$ remains independent of $u_{1,2}$. We have $|P_3 - P_2| \leq 1/p$.

Game 4: In this game, if the adversary $\mathcal{A}$ outputs a *valid* ciphertext $\mathsf{CT}^* = (\mathbf{c}^*, \mathbf{ovk}^*, \boldsymbol{\sigma}^*_{\mathsf{trace}}, \boldsymbol{\pi}^*_{\mathsf{valid}})$ such that $\mathbf{ovk}^* = \mathbf{ovk}$, we use the trapdoor key τ_σ to extract a valid signature $\boldsymbol{\sigma}^*_1 = (R^*_1, S^*_1)$ from the commitments $\boldsymbol{C}_R, \boldsymbol{C}_S$. This is always possible since $\mathbf{crs}_\sigma$ is in the binding mode for group elements, and $\hat{\boldsymbol{\pi}}_{sig}$ is then perfectly sound. This signature $\boldsymbol{\sigma}^*_1 = (R^*_1, S^*_1)$ satisfies $e(R^*_1, \hat{g}) \cdot e(S^*_1, \hat{h}) = e(g, \hat{l}_1) \cdot e(c^*, \hat{l}_2) \cdot e(d^*, \hat{l}_3)$, where $\mathbf{ovk} = (\hat{l}_i)^3_{i=1}$ and $c^* = \prod^\ell_{i=1} c^*_i$ from $\mathbf{c}^* = (d^*_1, d^*_2, (c^*_i)^\ell_{i=1})$. Now, given the vector (g, c^*, d^*_1) we compute a fresh LHSP signature with $\mathbf{osk}$ resulting in $\boldsymbol{\sigma}^\dagger_1 = (R^\dagger_1, S^\dagger_1)$. In the event that $\boldsymbol{\sigma}^\dagger_1 \neq \boldsymbol{\sigma}^*_1$, we abort the game and output 0. If we let this event be $\mathcal{F}_4$, we have $|P_4 - P_3| \leq \Pr[\mathcal{F}_4]$.

It is easy to see that with two distinct valid LHSP signatures on the same vector we have $e(R^*_1/R^\dagger_1, \hat{g}) \cdot e(S^*_1/S^\dagger_1, \hat{h}) = 1_{\mathbb{G}_T}$. Given $\hat{g}, \hat{h} \in \hat{\mathbb{G}}$, computing these elements of $\mathbb{G}$ comes to break to SXDH assumption. We thus have $\Pr[\mathcal{F}_4] \leq \varepsilon_{\mathtt{sxdh}}$.

Game 5: This game brings a slight modification to $\mathcal{H}_4$: we generate $\mathbf{crs}_\pi = (\boldsymbol{\varphi}, \boldsymbol{\psi})$ such that $\boldsymbol{\varphi} = (\hat{f}\hat{\varphi}^\xi, \hat{\varphi})$ and $\boldsymbol{\psi} = (\hat{\psi}^\xi, \hat{\psi})$ for random $\hat{f}, \hat{\varphi}, \hat{\psi} \in \hat{\mathbb{G}}$. In the ROM variant, we again program the random oracle to output $(\mathbf{crs}_\sigma, \mathbf{crs}_\pi) \leftarrow \mathsf{Hash}(g, \hat{g}, (f_i)^\ell_{i=1})$ in the generation of PK. Eventually, we abort the game, and output 0 if $\hat{f} = 1_{\hat{\mathbb{G}}}$. As long as $\hat{\psi} \neq 1_{\hat{\mathbb{G}}}$, the first component of $\boldsymbol{\psi}$ remains independent as in $\mathcal{H}_4$. We have $|P_5 - P_4| \leq 2/p$.

Game 6: This game is as $\mathcal{H}_5$ except that we introduce a failure event $\mathcal{F}_6$ which causes the game to abort and outputs 0. $\mathcal{F}_6$ occurs when the adversary $\mathcal{A}$ outputs a *valid* ciphertext $\mathsf{CT}^* = (\mathbf{c}^*, \mathbf{ovk}^*, \boldsymbol{\sigma}^*_{\mathsf{trace}}, \boldsymbol{\pi}^*_{\mathsf{valid}})$ such that $\boldsymbol{\pi}^*_{\mathsf{valid}} = (\boldsymbol{\varphi}^*_0, \boldsymbol{\pi}^*_0, \boldsymbol{\pi}^*_1)$ with $\boldsymbol{\varphi}^*_0 = (\varphi^*_{01}, \varphi^*_{02})$ but $\varphi^*_{01} \neq \hat{f} \cdot (\varphi^*_{02})^\xi$.

We have $|P_6 - P_5| \leq \Pr[\mathcal{F}_6] = 1/p$. Indeed, since CT^* is valid we have $\boldsymbol{\varphi} = \boldsymbol{\varphi}^*_0 \boldsymbol{\varphi}^*_1$ and it suffices to prove that and $\boldsymbol{\varphi}^*_1 = \boldsymbol{\psi}^\gamma$ for some $\gamma \in \mathbb{Z}_p$, but with probability $1/p$. If there is no such γ, $\mathbf{crs}_1 = (\boldsymbol{\varphi}^*_1, \boldsymbol{\psi})$ would be in the binding mode to commit to scalars and the GS proofs $\boldsymbol{\pi}^*_1$ would be perfectly sound. However, the statement for (A, B) in PK is false since (g, h, A, B) is not a DH tuple except with probability $1/p$.

Note: since $\boldsymbol{\varphi}^*_0 = (\hat{f}(\varphi^*_{02})^\xi, \varphi^*_{02})$ and $\boldsymbol{\psi} = (\hat{\psi}^\xi, \hat{\psi})$, the CRS $\mathbf{crs}_0 = (\boldsymbol{\varphi}^*_0, \boldsymbol{\psi})$ is in the binding mode to commit to scalars and the GS proofs $\boldsymbol{\pi}^*_0$ is perfectly sound. Therefore, by computing $\mathsf{Dec}(\mathsf{SK}, \mathsf{CT}^*)$ we can always find a bit-string message $\mathbf{m}^* = (m^*_1, \ldots, m^*_\ell) \in \mathbb{Z}^\ell_p$ such that $\mathbf{c}^* = (d^*_1, d^*_2, (c^*_i)^\ell_{i=1})$ can be written as $c^*_i = g_i^{m^*_i} f_i^{\theta^*}$ for each $i \in [\ell]$, and $d_1 = g^{\theta^*}$ and $d_2 = h^{\theta^*}$, for some $\theta^* \in \mathbb{Z}_p$. That is, if $m_i = 1$, it is not only because $c^*_i \neq (d^*_1)^{\alpha_i}(d^*_2)^{\beta_i}$ as in the decryption algorithm, but because $c^*_i = g_i \cdot (d^*_1)^{\alpha_i}(d^*_2)^{\beta_i}$. (See item 2 in the security proof of the verifiability for more details.)

Game 7: In this game we bring yet another modification to the previous game $\mathcal{H}_6$. First, when we decrypt the bit-string message $\mathbf{m}^* = (m^*_1, \ldots, m^*_\ell) \in \mathbb{Z}^\ell_p$, we compute the group element $M^* = \prod^\ell_{i=1} g_i^{m^*_i}$. Then, we also compute the group element $M = \prod^\ell_{i=1} g_i^{m_i}$ given $\mathcal{A}$'s chosen message $\mathbf{m} = (m_1, \ldots, m_\ell) \in \mathbb{Z}^\ell_p$. Finally, we abort the game and output 0 if $M^* = M$ but $\mathbf{m}^* \neq \mathbf{m}$. If we call this latter event $\mathcal{F}_7$, we find $|P_7 - P_6| \leq \Pr[\mathcal{F}_7]$.

Clearly, if $\mathcal{F}_7$ occurs, we can build a reduction to DLog. That is because we can generate the components g_i of PK as $g_i = g^{\delta_i} h^{\chi_i}$ for random $\delta_i, \chi_i \leftarrow\!\$\, \mathbb{Z}_p$, for all $i \in [\ell]$. Assuming that $M^* = M$ but $\mathbf{m}^* \neq \mathbf{m}$, we can rewrite the equality as $g^{\sum_{i\in[\ell]} \delta_i(m^*_i - m_i)} = h^{\sum_{i\in[\ell]} \chi(m_i - m^*_i)}$. As long as $m^*_j \neq m_j$ for some $j \in [\ell]$, we have $\sum_{i\in[\ell]} \chi(m_i - m^*_i) \neq 0$, but with probability $1/p$. Therefore, $\Pr[\mathcal{F}_7] \leq 1/p + \varepsilon_{\mathtt{sxdh}}$.

Game 8: This game is like $\mathcal{H}_7$, but we bring a last modification. In the key generation algorithm of the standard model construction, we define $F = f^\mu$ and $G = g^\mu$ for a random $\mu \in \mathbb{Z}_p$, where $f = \prod^\ell_{i=1} f_i$ as in item 2 of the encryption algorithm. In the ROM variant, the challenger simply programs $(C, F, G) = \mathsf{Hash}(\hat{g}, \mathbf{ovk})$ with those parameters at the outset of the game. In both cases, we have $|P_8 - P_7| \leq \varepsilon_{\mathtt{sxdh}}$.

Indeed, given an SXDH instance (g, h, G, H), the distinguisher defines the public elements f_i as $f_i = g^{\alpha_i} h^{\beta_i}$ for all $i \in [\ell]$ by following the key generation, but it also computes $F_i = G^{\alpha_i} H^{\beta_i}$ for all $i \in [\ell]$ and sets $(F = \prod^\ell_{i=1} F_i, G)$ in both cases. Then, it finalizes the generation of PK and finally hands the adversary with $\mathsf{SK} = (\mathsf{sk}_i)^\ell_{i=1} = (\alpha_i, \beta_i)^\ell_{i=1}$ as in all the previous games. The remaining part of the reduction is trivial.

Conclusion. Now, we argue that $\Pr[\mathcal{H}_8 = 1] = P_8 \leq 1/p$. To see that, let assume that $\mathcal{A}$ wins in $\mathcal{H}_8$. From $\mathcal{H}_4$, we know that the extracted signature $\boldsymbol{\sigma}^*_1$ on (g, c^*, d^*_1), where $c^* = \prod^\ell_{i=1} c^*_i$ from $\mathbf{c}^* = (d^*_1, d^*_2, (c^*_i)^\ell_{i=1})$ is equal to the honestly computed one $\boldsymbol{\sigma}^\dagger_1$ from $\mathbf{osk}$. Moreover, during the whole game the adversary only sees two LHSP signatures related to $\mathbf{osk}$ on two independent vectors. That is because the committed signature

$\boldsymbol{\sigma}_1$ authenticates the vector $(g, c, d_1) = (g, \prod_{i=1}^{\ell} c_i, d_1) = (g, Mf^{\theta}, g^{\theta})$, where $M = \prod_{i=1}^{\ell} g_i^{m_i}$ (as defined from $\mathcal{H}_7$), $\boldsymbol{\sigma}_2$ authenticates the vector $(1, f, g)$, where $f = \prod_{i=1}^{\ell} f_i$, and $\boldsymbol{\sigma}_3 = \boldsymbol{\sigma}_2^{\mu}$ since the third vector $(1, F, G) = (1, f, g)^{\mu}$ from $\mathcal{H}_8$. Therefore, the LHSP signatures authenticate a subspace of dimension 2 in $\mathbb{G}^3$. To conclude, we show that the authenticated vector $(g, c^*, d_1^*) = (g, \prod_{i=1}^{\ell} c_i^*, d_1^*) = (g, M^* f^{\theta^*}, g^{\theta^*})$ of the adversary is outside the linear span generated by (g, c, d_1) and $(1, f, g)$, and then $\boldsymbol{\sigma}_1^{\dagger}$ remains unpredictable, but with probability $1/p$. It suffices to show that $(g, M^*, 1)$ is not a linear combination of $(g, M, 1)$ and $(1, f, g)$. But, this is obvious since $M^* \neq M$ from $\mathcal{H}_7$. So, $\boldsymbol{\sigma}_1^{\dagger} = (g^{\eta_1}(c^*)^{\eta_2}(d_1^*)^{\eta_3}, g^{\zeta_1}(c^*)^{\zeta_2}(d_1^*)^{\zeta_3})$ is independent of $\mathcal{A}$'s view and remains uniform (from linear algebra).

All in all, in both constructions we have $\varepsilon_{\texttt{trace}} \leq 4 \cdot \varepsilon_{\texttt{sxdh}} + 7/p$. □

*Proof (**(Almost-Tight) TCCA security of Theorem 1**).* Let $\mathcal{A}$ be an adversary against the mTCCA security of our TREnc, and $\varepsilon_{\texttt{tcca}}$ be its winning probability in the mTCCA game as defined in Figure 1 (right) of Definition 5. We give a succession of hybrid games $\mathcal{H}_0, \ldots, \mathcal{H}_{15}$, where $\mathcal{H}_0$ is the real mTCCA game. In each of those games, the challenger always know the secret key SK to answer any decryption queries appropriately. For each of those games, we denote P_i the probability that $\mathcal{H}_i$ outputs 1 in the security parameter λ.

Let q be the number of challenge queries and q_{dec} be the number of decryption challenge made by $\mathcal{A}$. For the standard model construction, we have $q = 1$. For simplicity, we talk about TCCA to refer to both the single-challenge version and the multi-challenge version. Let also q_{ovk} be the total number of Hash-queries of the form $(C, F, G) \leftarrow \mathsf{Hash}(\hat{g}, \mathbf{ovk})$ for any triple $\mathbf{ovk} \in \hat{\mathbb{G}}^3$ made by the aversary in the ROM-based TREnc construction.

Game 0: This is the real TCCA game where, the challenger runs $\mathsf{Gen}(1^{\lambda})$ to generate $(\mathsf{PK}, \mathsf{SK})$ with $\mathsf{pp} \leftarrow \mathsf{Setup}(1^{\lambda})$ and $\ell = \mathsf{poly}'(\lambda)$ such that $\mathsf{SK} = (\mathsf{sk}_i)_{i=1}^{\ell} = (\alpha_i)_{i=1}^{\ell} \leftarrow\!\!\$\, \mathbb{Z}_p^{\ell}$ and $\mathsf{PK} = (g, h, (g_i, f_i)_{i=1}^{\ell}, A, B, \mathbf{crs}_{\sigma}, \mathbf{crs}_{\pi}, \hat{g}, \hat{h})$, where $f_i = g^{\alpha_i}$, for each $i \in [\ell]$. In both constructions, $\mathbf{crs}_{\sigma} \in \mathbb{G}^4$ and $\mathbf{crs}_{\pi} \in \hat{\mathbb{G}}^4$ are uniform, meaning that $\mathbf{crs}_{\sigma}$ is in the perfectly hiding/WI mode for group elements in $\mathbb{G}$, and $\mathbf{crs}_{\pi}$ is in the perfectly binding/sound mode for scalars in $\mathbb{Z}_p$ (as long as it is not a DH tuple). The challenger hands the adversary with PK, and also flips a random bit $b \leftarrow\!\!\$\, \{0, 1\}$. At each decryption query, given CT_i the challenger checks whether its trace is in the list of the traces that already appeared in a challenge query (where for the standard model construction, this list is either empty or a singleton). If so, the challenger outputs $\perp$. If not, it sends $\mathsf{Dec}(\mathsf{SK}, \mathsf{CT}_i)$ to the adversary $\mathcal{A}$. At each challenge query, given $(\mathsf{CT}_j^0, \mathsf{CT}_j^1)$ the challenger checks whether both ciphertexts are valid and share the same trace, and if this trace never appeared in a previous challenge query. If so, we say that the challenge query is *valid*. In that case, the challenger adds the common trace, denoted $\mathbf{ovk}_j^*$, to the list L of challenge traces, computes $\mathsf{CT}_j^* \leftarrow \mathsf{Rand}(\mathsf{PK}, \mathsf{CT}_j^b)$, and sends CT_j^* to $\mathcal{A}$. If the query is invalid, it simply outputs $\perp$. At the end of the game, $\mathcal{A}$ outputs b', and the challenger outputs 1 if $b = b$, and 0, otherwise. This latter bit is the output of $\mathcal{H}_0$. By definition, $P_0 = \varepsilon_{\texttt{tcca}}$.

In the next games, we assume that the challenger programs the random oracle and consistently replies to repeating Hash-queries in the ROM case.

Game 1: In this game, the challenger exactly follows $\mathcal{H}_0$ except in the ROM variant. In this case, it directly outputs $\perp$ if the adversary makes a challenge query $(\mathsf{CT}_j^0, \mathsf{CT}_j^1)$ with a common trace $\mathbf{ovk}_j^*$ but for which there is still no entry in the Hash-list. That is, by emulating the random oracle, the challenger never defined an input-output pair for $(\hat{g}, \mathbf{ovk}_j^*)$. Since, when defining $(C_j^*, F_j^*, G_j^*) \leftarrow \mathsf{Hash}(\hat{g}, \mathbf{ovk}_j^*)$ in the ROM variant, there is a single group element C_j^* that fulfills the last verification equation of item 6 in the verification procedure, the probability of getting a valid proof π_1 is thus bounded by $1/p$. Therefore, $|P_1 - P_0| \leq q/p$ in the ROM variant, and $P_1 = P_0$ otherwise.

Without loss of generality, from now on we assume that all the q challenge queries are valid, which also means that the output (C_j^*, F_j^*, G_j^*) of Hash in the ROM variant on input $(\hat{g}, \mathbf{ovk}_j^*)$ is already defined prior to the challenge query $(\mathsf{CT}_j^0, \mathsf{CT}_j^1)$.

Game 2: This game is as $\mathcal{H}_1$ except that the challenger aborts the game and outputs a random bit if the adversary managed to make a valid challenge query on input $(\mathsf{CT}_j^0, \mathsf{CT}_j^1)$ such that both ciphertexts are valid for their common trace $\mathbf{ovk}_j^*$, but the valid LHSP signatures $\boldsymbol{\sigma}_2^{j,0}, \boldsymbol{\sigma}_3^{j,0} \in \boldsymbol{\sigma}_{\mathsf{trace}}^{j,0}$ and $\boldsymbol{\sigma}_2^{j,1}, \boldsymbol{\sigma}_3^{j,1} \in \boldsymbol{\sigma}_{\mathsf{trace}}^{j,1}$ on the last two (common) rows of their respective matrix $\mathbf{T}_j^0$ and $\mathbf{T}_j^1$ (as recomputed in item 2 of the verification algorithm) are such that $\boldsymbol{\sigma}_2^{j,0} \neq \boldsymbol{\sigma}_2^{j,1}$ or $\boldsymbol{\sigma}_3^{j,0} \neq \boldsymbol{\sigma}_3^{j,1}$. Since these LHSP signatures $\boldsymbol{\sigma}_2^{j,0}, \boldsymbol{\sigma}_2^{j,1}$ and $\boldsymbol{\sigma}_3^{j,0}, \boldsymbol{\sigma}_3^{j,1}$ are valid respectively on $(1, f, g)$ and either $(1, F, G)$ or $(1, F_j^*, G_j^*)$ in the ROM variant, (where $(C_j^*, F_j^*, G_j^*) \leftarrow \mathsf{Hash}(\hat{g}, \mathbf{ovk}_j^*)$) for both ciphertexts, the challenger gets two distinct signatures on the same vector.

Since the validity of the LHSP signatures are all related to the same parameter $(\hat{g}, \hat{h})$ for all the $\mathbf{ovk}_j^*$ and the hardness of SXDH implies the infeasibility of producing two distinct LHSP signatures on the same vectors, $|P_2 - P_1| \leq \varepsilon_{\mathtt{sxdh}}$.

Game 3: For the standard model construction, this game is as $\mathcal{H}_2$ except that the challenger aborts the game and outputs a random bit if the adversary outputs two different ciphertexts containing distinct $\mathbf{ovk}$ and $\mathbf{ovk}'$ such that $\mathsf{Hash}(\hat{g}, \mathbf{ovk}) =: \tau = \tau' := \mathsf{Hash}(\hat{g}, \mathbf{ovk}')$. Clearly, this would contradict the collision-resistance of Hash and thus $|P_3 - P_2| \leq \varepsilon_{\mathtt{cr}}$.

For the ROM variant, this game is as $\mathcal{H}_2$ except that the challenger aborts the game and outputs a random bit if, when the challenger defines a new output (C, F, G) to answer a Hash-query on input a new $(\hat{g}, \mathbf{ovk})$, C is already the first component of another output. That is, the challenger directly aborts if there is a collision on the first component C among the triples (C, F, G) programmed as the outputs of Hash on input of the form $(\hat{g}, \mathbf{ovk}) \in \hat{\mathbb{G}}^4$. Obviously, $|P_3 - P_2| \leq q_{\mathsf{ovk}}(q_{\mathsf{ovk}} - 1)/2p$.

Game 4: This game brings a slight modification to $\mathcal{H}_3$ in the way the challenger generates $\mathbf{crs}_\pi = (\boldsymbol{\varphi}, \boldsymbol{\psi})$ in PK. Instead of picking $\mathbf{crs}_\pi$ uniformly at random over $\hat{\mathbb{G}}^4$, it now picks $\boldsymbol{\psi} \leftarrow\!\!\$\, \hat{\mathbb{G}}^2$ and $\nu \leftarrow\!\!\$\, \mathbb{Z}_p$, and computes $\boldsymbol{\varphi} = \boldsymbol{\psi}^\nu$. Then, in the ROM variant it programs the random oracle to output $(\mathbf{crs}_\sigma, \mathbf{crs}_\pi) \leftarrow \mathsf{Hash}(g, \hat{g}, (f_i)_{i=1}^\ell)$ in the key generation.

In $\mathcal{H}_3$, $\mathbf{crs}_\pi$ was in the binding mode while now it is in the hiding mode (allowing simulation). Obviously, $|P_4 - P_3| \leq \varepsilon_{\mathtt{sxdh}} + 1/p$.

Game 5: In this game, the challenger still honestly randomizes the CPA encryption part $\mathbf{c}_j^b$ and the tracing part $\boldsymbol{\sigma}_{\mathsf{trace}}^{j,b}$ of $\mathsf{CT}_j^b = (\mathbf{c}_j^b, \mathbf{ovk}_j^*, \boldsymbol{\sigma}_{\mathsf{trace}}^{j,b}, \boldsymbol{\pi}_{\mathsf{valid}}^{j,b})$ in the j-th valid challenge query, but it computes a simulated proof $\boldsymbol{\pi}_{\mathsf{valid}}^{j,*}$ from scratch instead of faithfully randomizing $\boldsymbol{\pi}_{\mathsf{valid}}^{j,b}$. To do so, the challenger computes $\boldsymbol{\varphi}_0^{j,*}$ afresh as $\boldsymbol{\varphi}_0^{j,*} = \boldsymbol{\psi}^{\gamma_j *}$ for random $\gamma_j^* \leftarrow\!\!\$\, \mathbb{Z}_p$, and sets $\mathbf{crs}_0^{j,*} = (\boldsymbol{\varphi}_0^{j,*}, \boldsymbol{\psi})$. Then, it derives the complement $\mathbf{crs}_1^{j,*} = (\boldsymbol{\varphi}_1^{j,*}, \boldsymbol{\psi})$ so that $\boldsymbol{\varphi}_0^{j,*}\boldsymbol{\varphi}_1^{j,*} = \boldsymbol{\varphi}$ as in the item 4 of the encryption algorithm, but with a simulation bit $b_{\mathsf{sim}} = 0$. Since $\boldsymbol{\varphi}_1^{j,*} = \boldsymbol{\psi}^{-\gamma_j * + \nu}$, the challenger can compute Groth-Sahai simulated proofs $\pi_0^{j,*}$ and $\pi_1^{j,*}$ for both branches of the OR-proof from the respective trapdoors γ_j^* and $-\gamma_j^* + \nu$.

As long as for all $j \in [q]$ such that $\boldsymbol{\varphi}_1^{j,0}$ (resp. $\boldsymbol{\varphi}_1^{j,1}$) can be written as $\boldsymbol{\psi}^{\gamma_{j,0}}$ (resp. $\boldsymbol{\psi}^{\gamma_{j,1}}$) for some $\gamma_{j,0} \in \mathbb{Z}_p$ (resp. $\gamma_{j,0} \in \mathbb{Z}_p$), both games are perfectly indistinguishable. Let $\mathcal{F}_5$ be the event that there is $\boldsymbol{\varphi}_1^{j,0}$ or $\boldsymbol{\varphi}_1^{j,1}$ not in the span generated by $\boldsymbol{\psi}$. We thus have, $|P_5 - P_4| \leq \Pr[\mathcal{F}_5]$. Next, we argue that as long as (A, B) in the public key (and derived from Hash in the ROM-based construction) is not in the span generated by (g, h), which is always the case but with probability $1/p$, this event never happens, i.e. $\Pr[\mathcal{F}_5] = 1/p$.

Indeed, if (A, B) cannot be written as $(g, h)^z$, for some $z \in \mathbb{Z}_p$, then no CRS $\mathbf{crs}_1^{j,0} = (\boldsymbol{\varphi}_0^{j,0}, \boldsymbol{\psi})$ (resp. $\mathbf{crs}_1^{j,1} = (\boldsymbol{\varphi}_0^{j,1}, \boldsymbol{\psi})$) from $\boldsymbol{\pi}_{\mathsf{valid}}^{j,0}$ (resp. $\boldsymbol{\pi}_{\mathsf{valid}}^{j,1}$) can be in the binding mode because the proof $\pi_1^{j,0}$ (resp. $\pi_1^{j,1}$) would be perfectly sound whereas the statement is false. To summarize, $|P_5 - P_4| \leq 1/p$.

Game 6: This game is as $\mathcal{H}_5$ except that given $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ the challenger picks a random $h \leftarrow\!\!\$\, \mathbb{G}$ and a random scalar $z \leftarrow\!\!\$\, \mathbb{Z}_p$ and computes $(A, B) = (g, h)^z$ during the key generation so that the statement about (A, B) in the OR-proof becomes true, where $g \in \mathsf{pp}$. In the ROM variant, it then programs the random oracle such that $A, B, (g_i)_{i=1}^\ell, h, \hat{h} \leftarrow \mathsf{Hash}(g, \hat{g}, \ell)$. The remaining part of $\mathcal{H}_6$ is as $\mathcal{H}_5$. Clearly, $|P_6 - P_5| \leq \varepsilon_{\mathtt{sxdh}} + 1/p$

Game 7: For the standard model construction, instead of generating C_0, C_1 uniformly at random directly from $\mathbb{G}^2$, the challenger picks $x_0, x_1, y_0, y_1 \leftarrow\!\!\$\, \mathbb{Z}_p$ and computes $C_0 = g^{x_0}h^{y_0}$ and $C_1 = g^{y_1}h^{y_1}$.

Analogously in the ROM variant, for all the hash queries of the form $\mathsf{Hash}(\hat{g}, \mathbf{ovk})$ with $\mathbf{ovk} \in \hat{\mathbb{G}}^3$, the challenger still picks $F, G \leftarrow\!\!\$\, \mathbb{G}$ as before but also picks random $x, y \leftarrow\!\!\$\, \mathbb{Z}_p$ and computes $C = g^x h^y$. Then, it programs $(C, F, G) \leftarrow \mathsf{Hash}(\hat{g}, \mathbf{ovk})$.

Since all the so-generated triples are still uniformly distributed as long as $g \neq 1$ or $h \neq 1$, we have $|P_7 - P_6| \leq 1/p$.

Game 8: In this game the challenger changes the way it answers to the challenge queries. Instead of simulating $\boldsymbol{\pi}_{\mathsf{valid}}^{j,*}$ from scratch as done from $\mathcal{H}_5$ until now, it computes $\boldsymbol{\pi}_{\mathsf{valid}}^{j,*}$ as a honest run of the OR-proof but for the statement (A, B, C_j^*), where either $C_j^* = C_0^{\tau_j^*} C_1 = g^{x_0\tau_j^*+x_1} h^{y_0\tau_j^*+y_1}$ for $\tau_j^* \leftarrow \mathsf{Hash}(\hat{g}, \mathbf{ovk}_j^*)$ in the standard model construction or $C_j^* = g^{x_j^*} h^{y_j^*}$ from $(C_j^*, F_j^*, G_j^*) \leftarrow \mathsf{Hash}(\hat{g}, \mathbf{ovk}_j^*)$ in the ROM variant. That is, the challenger still no more use $\boldsymbol{\pi}_{\mathsf{valid}}^{j,b}$, but when it computes $\boldsymbol{\varphi}_0^{j,*} = \boldsymbol{\psi}^{\gamma_j *}$ and $\boldsymbol{\varphi}_1^{j,*}$ such that $\boldsymbol{\varphi} = \boldsymbol{\varphi}_0^{j,*}\boldsymbol{\varphi}_1^{j,*}$, it no more uses the simulation trapdoor ν introduced in $\mathcal{H}_4$. Therefore, the challenger uses γ_j^* to compute a simulated $\pi_0^{j,*}$ from $\mathbf{crs}_0^{j,*}$ as before, but it computes an honest proof $\pi_1^{j,*}$ from $\mathbf{crs}_1^{j,*}$ with the witness (x_j^*, y_j^*, z) satisfying $(A, B) = (g, h)^z$ and $C_j^* = g^{x_j^*} h^{y_j^*}$. The remaining parts of the answers to valid challenge queries are addressed as in $\mathcal{H}_7$. In particular, $\mathbf{c}_j^b$ and $\boldsymbol{\sigma}_{\mathsf{trace}}^{j,b}$ are still honestly adapted and randomized. Answers to decryption queries are unchanged.

Since $\mathbf{crs}_\pi$ is still in the hiding mode so are $\mathbf{crs}_0^{j,*}$ and $\mathbf{crs}_1^{j,*}$ as in $\mathcal{H}_7$, even if ν is useless in this game. In that case, the proof parts $\boldsymbol{\pi}_{\mathsf{valid}}^{j,*}$ are distributed exactly as before since they are still in the perfect witness indistinguishable mode, which leads to $P_8 = P_7$.

Game 9: In this game the challenger removes the modification introduced in $\mathcal{H}_4$: it truns back $\mathbf{crs}_\pi = (\boldsymbol{\varphi}, \boldsymbol{\psi})$ in the binding mode. More precisely, it honestly generate uniform $(\mathbf{crs}_\sigma, \mathbf{crs}_\pi) \leftarrow\!\!\$\ \mathbb{G}^4 \times \hat{\mathbb{G}}^4$ as in $\mathsf{Gen}(1^\lambda)$ and programs $\mathsf{Hash}(g, \hat{g}, (f_i)_{i=1}^\ell)$ to this value in the ROM variant. Thus, $|P_9 - P_8| \le \varepsilon_{\mathtt{sxdh}} + 1/p$.

Game 10: The challenger brings a slight modification in the way $\mathbf{crs}_\pi = (\boldsymbol{\varphi}, \boldsymbol{\psi})$ is generated in the binding mode. During the key generation, it picks random $\hat{f}, \hat{\varphi}, \hat{\psi} \leftarrow\!\!\$\ \hat{\mathbb{G}}$ and $\xi \leftarrow\!\!\$\ \mathbb{Z}_p$, and defines $\boldsymbol{\varphi} = (\hat{f}\hat{\varphi}^\xi, \hat{\varphi})$ and $\boldsymbol{\psi} = (\hat{\psi}^\xi, \hat{\psi})$. It then programs again the random oracle such that $(\mathbf{crs}_\sigma, \mathbf{crs}_\pi) \leftarrow \mathsf{Hash}(g, \hat{g}, (f_i)_{i=1}^\ell)$ for the ROM-based construction. As long as $\hat{\psi} \neq 1_{\hat{\mathbb{G}}}$, the first component of $\boldsymbol{\psi}$ remains independent as in $\mathcal{H}_9$. We have $|P_{10} - P_9| \le 1/p$.

Game 11: In this game the challenger changes the way it answers to the decryption and challenge queries. On *non-aborting valid* input $\mathsf{CT} = (\mathbf{c}, \mathbf{ovk}, \boldsymbol{\sigma}_{\mathsf{trace}}, \boldsymbol{\pi}_{\mathsf{valid}})$ queried for decryption, where $\boldsymbol{\pi}_{\mathsf{valid}} = (\boldsymbol{\varphi}_0, \boldsymbol{\pi}_0, \boldsymbol{\pi}_1)$ and $\boldsymbol{\varphi}_0 = (\hat{\varphi}_{01}, \hat{\varphi}_{02})$, it outputs $\perp$ if $\hat{\varphi}_{01} \neq \hat{f} \cdot \hat{\varphi}_{02}^\xi$. Similarly, on *non-aborting valid* input $(\mathsf{CT}_j^0, \mathsf{CT}_j^1)$ in a challenge query, the challenger makes the analogue checks on both ciphertexts and returns $\perp$ if at least one inequality occurs. If we call this latter event $\mathcal{F}_{11}$, we clearly have $|P_{11} - P_{10}| = \Pr[\mathcal{F}_{11}]$.

Now, we argue that $\Pr[\mathcal{F}_{11}] \le \varepsilon_{\mathtt{sxdh}} + (q_{\mathsf{dec}} + 2q)/p$. Indeed, since $\boldsymbol{\varphi} = (\hat{f}\hat{\varphi}^\xi, \hat{\varphi})$ from $\mathcal{H}_{10}$ by construction of $\mathbf{crs}_\pi$ in PK and $\boldsymbol{\varphi} = \boldsymbol{\varphi}_0\boldsymbol{\varphi}_1$ by the validity of CT, we know that $1_{\mathbb{G}} \neq \hat{G} := \hat{\varphi}_{11}/\hat{\varphi}_{12}^\xi$, where $\boldsymbol{\varphi}_1 = (\hat{\varphi}_{11}, \hat{\varphi}_{12})$, if $\mathcal{F}_{11}$ occurs. In that case, the perfectly sound proof $\boldsymbol{\pi}_1 = (\hat{\boldsymbol{c}}_{\mathsf{enc}}, \boldsymbol{\pi}_{\mathsf{enc}}, \hat{\boldsymbol{c}}_{\mathsf{com}}, \pi_{\mathsf{com}})$ is extractable with ξ. Let us focus on the proof of opening $\hat{\boldsymbol{c}}_{\mathsf{com}}$ and π_{com} of C, where C is computed from $\mathsf{Hash}(\hat{g}, \mathbf{ovk})$ and is $C = g^x h^y$ for $x = \tau x_0 + x_1$ and $y = \tau y_0 + y_1$ for the standard model construction, or programmed as $C = g^x h^y$ in the ROM variant. From $\mathcal{H}_7$ and the collision-resistance of C from $\mathcal{H}_3$, it is easy to build a Double-Pairing adversary given the parameter (g, h) that are common to all openings. The extracted X, Y from $\hat{\boldsymbol{c}}_{\mathsf{com}}$ satisfy $e(C, \hat{G}) = e(g, X)e(h, Y)$. However, $(X, Y) \neq (\hat{G}^x, \hat{G}^y)$ but with negligible probability $1/p$. That is because, either the decryption query for CT happens before a challenge query that contain the same trace $\mathbf{ovk}$ or it is never the case throughout the game, or CT is queried for decryption after a challenge query with the same trace. Since by assumption CT is valid the latter case does not occur (since such "post-challenge" query is forbidden in the TCCA game, so from $\mathcal{H}_0$). However, in the former case the TCCA adversary tries to open the perfectly hiding commitment C for which no additional information are given at that time. So, the first time it manages to open C it must be with another opening than $(\hat{G}^x, \hat{G}^y)$ except with probability $1/p$. The same argument holds for both ciphertexts in any valid non-aborting challenge query. Finally, if $(X, Y) \neq (\hat{G}^x, \hat{G}^y)$ holds, then the DP adversary outputs $(X/\hat{G}^x, Y/\hat{G}^y)$ and always wins. Since a DP adversary implies an SXDH adversary, the result follows.

Game 12: In this game we bring yet another modification in the way the challenger answers to the decryption queries. Namely, on the input ciphertext CT queried for decryption by the adversary, the challenger proceeds as before to determine the validity of CT but it no more uses SK to compute the plaintext from the CPA part $\mathbf{c} = (d_1, d_2, (c_i)_{i=1}^\ell)$. Instead, if CT is deemed valid and does not lead to an early abort, the challenger uses the trapdoor extracting key ξ to compute $\mathbf{m} = (m_i)_{i=1}^\ell$ from $\boldsymbol{\pi}_0 = (\boldsymbol{\pi}_{\mathsf{ciph}}, \hat{\boldsymbol{c}}_{\mathsf{ciph}})$ in $\boldsymbol{\pi}_{\mathsf{valid}} = (\boldsymbol{\varphi}_0, \boldsymbol{\pi}_0, \boldsymbol{\pi}_1)$ as follows. On input ξ and the commitments $(\boldsymbol{c}_i)_{i=1}^\ell$ in $\hat{\boldsymbol{c}}_{\mathsf{ciph}}$, it extracts the group elements $\hat{\boldsymbol{M}} = (\hat{M}_i)_{i=1}^\ell$, and if $\hat{M}_i = 1_{\hat{\mathbb{G}}}$, the challenger sets $m_i = 0$, else it sets 1.

Now we show that $P_{12} = P_{11}$. Indeed, non-rejecting CT from $\mathcal{H}_{11}$ are such that $\hat{\varphi}_{01} = \hat{f}\hat{\varphi}_{02}^\xi$ and then the perfectly sound proof $\boldsymbol{\pi}_{\mathsf{ciph}}$ ensures that $\hat{M}_i = \hat{f}^{m_i}$ for all $i \in [\ell]$, where these m_i's underlie the CPA part

$\mathbf{c} = (d_1, d_2, (c_i)_{i=1}^{\ell})$ that must be of the form $\mathbf{c} = (g^{\theta}, h^{\theta}, (g_i^{m_i} f_i^{\theta})_{i=1}^{\ell})$ for some $\theta \in \mathbb{Z}_p$ since equation 6 holds, and such that $m_i \in \{0,1\}$ since equations 7 holds as well. In other words, $\boldsymbol{\pi}_{\mathsf{valid}} = (\boldsymbol{\varphi}_0, \boldsymbol{\pi}_0, \boldsymbol{\pi}_1)$ is a perfectly sound proof that $\mathbf{c}$ is well-formed, and ξ always allows extracting the witness $\mathbf{m} = (m_i)_{i=1}^{\ell}$.

Note: despite this above extraction, we recall that in the answers to any valid non-aborting challenge query $(\mathsf{CT}_j^0, \mathsf{CT}_j^1)$, the ciphertext CT_j^* is computed from a simulated proof from $\mathcal{H}_8$ and such that $\hat{\varphi}_{11} = \hat{f}\hat{\varphi}_{12}^{\xi}$. These simulated proofs are still computed from a true statement, but by honestly proving the OR-proof by using the programmed witness for C_j^* generated from $\mathsf{Hash}(\hat{g}, \mathbf{ovk}_j^*)$ instead of randomizing the valid adversarially generated proof $\boldsymbol{\pi}_{\mathsf{valid}}^{j,b}$.

Game 13: This game deviates from the previous one by the way the challenger computes (F, G) in PK for the standard model construction, and in each of the q_{ovk} answers to the Hash-queries of the form $\mathsf{Hash}(\hat{g}, \mathbf{ovk})$ for any triple $\mathbf{ovk}$ in $\hat{\mathbb{G}}^3$ for the ROM-based variant. While C is still programmed as in $\mathcal{H}_7$, the challenger now computes (F, G) as a random vector in the linear span generated by (f, g), where $f = \prod_{i=1}^{\ell} f_i$ (as in the previous games), instead of drawing it uniformly at random from $\mathbb{G}^2$. To do so, the challenger simply picks a random $w \hookleftarrow\!\!\$\, \mathbb{Z}_p$, and computes $(F, G) = (f, g)^w$ in PK for the standard model construction, and computes $(F, G) = (f, g)^w$ with a fresh exponent before programming $(C, F, G) \leftarrow \mathsf{Hash}(\hat{g}, \mathbf{ovk})$ in the ROM-based variant. For the standard model construction, we have $|P_{13} - P_{12}| \leq \varepsilon_{\mathtt{sxdh}} + 1/p$. Next, we show that $|P_{13} - P_{12}| \leq \varepsilon_{\mathtt{sxdh}} + 3/p$ for the ROM-based variant.

To see why a single reduction to SXDH works for all the Hash-queries of this form, we observe that they can all be answered given a single SXDH challenge (g, f, G_0, F_0), where g is used to generate the public parameters of PK as in both games $\mathcal{H}_{12}$ and $\mathcal{H}_{13}$, but for all $i \in [\ell]$, f_i is computed as $f_i = g_i^{\alpha}$ for $\alpha_i \hookleftarrow\!\!\$\, \mathbb{Z}_p$ if $i > 1$, and $f_1 = f / \prod_{i=2}^{\ell}$. That means that sk_1 is unknown even if PK as the same distribution in both games. Now, to answer to each fresh Hash-query on input $(\hat{g}, \mathbf{ovk})$, the reduction simply picks fresh $u, v \hookleftarrow\!\!\$\, \mathbb{Z}_p$, computes $G = g^u G_0^v$ and $F = f^u F_0^v$, and programs $(C, F, G) \leftarrow \mathsf{Hash}(\hat{g}, \mathbf{ovk})$ with C as from $\mathcal{H}_7$.

Assuming that $g, f \neq 1_{\mathbb{G}}$, let $(G_0, F_0) = (g, f)^{\kappa}(1, g^{\upsilon})^{\beta}$ for some unknown random $\kappa, \upsilon \in \mathbb{Z}_p$ and $\beta \in \{0, 1\}$. Then, the pair (F, G) in the output of any Hash-query can be written as $(F, G) = (f, g)^{u+\kappa v} \cdot (g^{\upsilon v}, 1)^{\beta}$, where $g^{\upsilon v}$ is a random element independent to the remaining view even given $w = u + \kappa v$ conditioned on $\upsilon \neq 0$. Therefore, if $\beta = 1$ the view is the one of the adversary in $\mathcal{H}_{12}$, and if $\beta = 0$ the view is the one of the adversary in $\mathcal{H}_{13}$.

Game 14: This game is as $\mathcal{H}_{13}$ except that the challenger modifies the way it answers to the valid non-aborting challenge queries. Given such $(\mathsf{CT}_j^0, \mathsf{CT}_j^1)$, the challenger retrieves w_j^* that defines $(F_j^*, G_j^*) = (f, g)^{w_j^*}$ in the answer to the Hash-query on input $(\hat{g}, \mathbf{ovk}_j^*)$ from the modification introduced in $\mathcal{H}_{13}$ (or in the public key for the standard model construction). Then, the challenger computes the intermediate values $(G_j^*, H_j^*, (F_{i,j}^*)_{i=1}^{\ell}) = (g, h, (f_i)_{i=1}^{\ell})^{w_j^*}$ satisfying $F_j^* = f^{w_j^*} = \prod_{i=1}^{\ell} f_i^{w_j^*} = \prod_{i=1}^{\ell} F_{i,j}^*$. Next, instead of randomizing the CPA part $\mathbf{c}_j^b = (d_1^{j,b}, d_2^{j,b}, (c_i^{j,b})_{i=1}^{\ell})$ and the tracing part $\boldsymbol{\sigma}_{\mathsf{trace}}^{j,b} = (\boldsymbol{C}_R^{j,b}, \boldsymbol{C}_S^{j,b}, \boldsymbol{\sigma}_2^{j,b}, \boldsymbol{\sigma}_3^{j,b}, \hat{\boldsymbol{\pi}}_{sig}^{j,b})$ of the ciphertext CT_j^b according to the real randomizing algorithm, the challenger now conducts the following steps. First, it picks random $\theta'_j, \theta''_j \hookleftarrow\!\!\$\, \mathbb{Z}_p$ and computes

$$\begin{aligned} \mathbf{c}_j^* &= \left(d_1^{j,*}, d_2^{j,*}, (c_i^{j,*})_{i=1}^{\ell}\right) \\ &= \left(d_1^{j,b}, d_2^{j,b}, (c_i^{j,b})_{i=1}^{\ell}\right) \cdot \left(g, h, (f_i)_{i=1}^{\ell}\right)^{\theta'_j} \cdot \left(G_j^*, H_j^*, (F_{i,j}^*)_{i=1}^{\ell}\right)^{\theta''_j}. \end{aligned} \tag{11}$$

That is, with respect to item 1 of the randomizing algorithm, it uses the additional vector $(G_j^*, H_j^*, (F_{i,j}^*)_{i=1}^{\ell})$ of intermediate values as a support to introduce the additional randomness θ''_j. Second, it adapts the commitments $\boldsymbol{C}_R^{j,b}, \boldsymbol{C}_S^{j,b}$ into

$$\begin{aligned} \tilde{\boldsymbol{C}}_R^{j,b} &= \boldsymbol{C}_R^{j,b} \cdot \iota(R_2^{j,b})^{\theta'} \cdot \iota(R_3^{j,b})^{\theta''} \\ \tilde{\boldsymbol{C}}_S^{j,b} &= \boldsymbol{C}_S^{j,b} \cdot \iota(S_2^{j,b})^{\theta'} \cdot \iota(S_3^{j,b})^{\theta''}, \end{aligned} \tag{12}$$

where, with respect to the item 2 of the randomizing algorithm, it not only uses $\boldsymbol{\sigma}_2^{j,b} = (R_2^{j,b}, S_2^{j,b})$ to adapt the committed signature but also $\boldsymbol{\sigma}_3^{j,b} = (R_3^{j,b}, S_3^{j,b})$. In other words, if $\boldsymbol{\sigma}_1^{j,b}$ was the committed LHSP signature relative to the first row of the matrix $\mathbf{T}_j^b$, the adapted LHSP signature would be $\boldsymbol{\sigma}_1^{j,*} = \boldsymbol{\sigma}_1^{j,b}(\boldsymbol{\sigma}_2^{j,b})^{\theta'}(\boldsymbol{\sigma}_3^{j,b})^{\theta''}$.

Finally, the challenger follows the steps of item 3 by honestly randomizing the commitments and the proof $\hat{\boldsymbol{\pi}}_{sig}^{j,b}$ leading to $\boldsymbol{\sigma}_{\mathsf{trace}}^{j,*} = (\boldsymbol{C}_R^{j,*}, \boldsymbol{C}_S^{j,*}, \boldsymbol{\sigma}_2^{j,*}, \boldsymbol{\sigma}_3^{j,*}, \hat{\boldsymbol{\pi}}_{sig}^{j,*})$, where $\boldsymbol{\sigma}_2^{j,*} = \boldsymbol{\sigma}_2^{j,b}$ and $\boldsymbol{\sigma}_3^{j,*} = \boldsymbol{\sigma}_3^{j,b}$. The validity proof part $\boldsymbol{\pi}_{\mathsf{valid}}^{j,*}$ is computed exactly as in $\mathcal{H}_{13}$.

Since the committed $\boldsymbol{\sigma}_1^{j,b}$ would be an LHSP signature on the first row $(g, \prod_{i=1}^{\ell} c_i^{j,b}, d_1^{j,b})$ of $\mathbf{T}_j^b$, together with $\boldsymbol{\sigma}_2^{j,b}$ on the second row $(1, f, g)$ and $\boldsymbol{\sigma}_3^{j,b}$ on the last row $(1, F_j^*, G_j^*)$, $\boldsymbol{\sigma}_1^{j,*}$ would be an LHSP signature on

$$\left(g, \textstyle\prod_{i=1}^{\ell} c_i^{j,b}, d_1^{j,b}\right) \cdot \left(1, f, g\right)^{\theta'_j} \cdot \left(1, F_j^*, G_j^*\right)^{\theta''_j}$$

that is nothing but the vector $(g, \prod_{i=1}^{\ell} c_i^{j,*}, d_1^{j,*})$ computed from $\mathbf{c}_j^*$. Moreover, due to the definition of the intermediate tuple $(G_j^*, H_j^*, (F_{i,j}^*)_{i=1}^{\ell})$ and that $(1, F_j^*, G_j^*) = (1, f, g)^{w_j^*}$ from $\mathcal{H}_{13}$, the CPA part satisfies

$$\mathbf{c}_j^* = \left(d_1^{j,b}, d_2^{j,b}, (c_i^{j,b})_{i=1}^{\ell}\right) \cdot \left(g, h, (f_i)_{i=1}^{\ell}\right)^{\theta'_j + w_j^* \theta''_j}.$$

Therefore, since crs_σ is in the hiding mode, the perfect witness indistinguishable proof $\hat{\boldsymbol{\pi}}_{sig}^{j,*}$ is distributed as an honest proof of knowledge of the signature $\boldsymbol{\sigma}_1^{j,b} (\boldsymbol{\sigma}_2^{j,b})^{\theta' + w_j^* \theta''}$ on the vector $(g, \prod_{i=1}^{\ell} c_i^{j,*}, d_1^{j,*})$ that is distributed exactly as in $\mathcal{H}_{13}$. Therefore, $P_{14} = P_{13}$.

Game 15: In this game, the challenger follows $\mathcal{H}_{14}$ except in the way the uniform public key elements $h, f, \ldots, f_\ell \in \mathbb{G}$ are generated. Now, it picks random scalars $\chi, \alpha_1, \ldots, \alpha_\ell \hookleftarrow\!\$\, \mathbb{Z}_p$, and computes h^χ, $f_1 = g^{\alpha_1}, \ldots, f_\ell = g^{\alpha_\ell}$. It then programs PK in the obvious way and sets SK as the empty string. Since SK of $\mathcal{H}_{14}$ was only used to generate these public key elements, the adversary's view is unchanged. We have $P_{15} = P_{14}$.

Game 16: In this game, we bring a last modification in how the challenger answers to the challenge query in the standard model case, and how it answers to the q_{ovk} Hash-queries and to the q challenge queries in the ROM case. First: (1) In the standard model, the challenger picks and stores $G, F_1, \ldots, F_\ell \hookleftarrow\!\$\, \mathbb{G}$ during the key generation, and set $F = \prod_{i=1}^{\ell} F_i$ and $H = G^\chi$; (2) In the ROM variant, for the Hash-queries on input $(\hat{g}, \mathbf{ovk})$ with a fresh $\mathbf{ovk} \in \hat{\mathbb{G}}^3$, the challenger picks and stores random intermediate values $G, F_1, \ldots, F_\ell \hookleftarrow\!\$\, \mathbb{G}$, computes $F = \prod_{i=}^{\ell} F_i$ and $H = G^\chi$, and programs $(C, F, G) \leftarrow \mathsf{Hash}(\hat{g}, \mathbf{ovk})$, where C is still computed as since $\mathcal{H}_7$. That is, $(1, F, G)$ is no more a random vector in the linear span generated by $(1, f, g)$, which cancels the change introduced in $\mathcal{H}_{13}$ and turns back $\mathbf{T}$ as a full rank with overwhelming probability. Second: (1) In the standard model, on input a valid non-aborting challenge query $(\mathsf{CT}^0, \mathsf{CT}^1)$ with common trace $\mathbf{ovk}^*$, the challenger retrieves the intermediate tuple $(G, H, (F_i)_{i=1}^{\ell})$ and uses it as a support the "over-randomize" $\mathbf{c}^b$ as in $\mathcal{H}_{14}$ with random $\theta', \theta'' \hookleftarrow\!\$\, \mathbb{Z}_p$; (2) In the ROM variant, on input a valid non-aborting challenge query $(\mathsf{CT}_j^0, \mathsf{CT}_j^1)$ with common trace $\mathbf{ovk}_j^*$, the challenger retrieves the intermediate tuple $(G_j^*, G_j^*, (F_{i,j}^*)_{i=1}^{\ell})$ generated to define (C_j^*, F_j^*, G_j^*) as the output of $\mathsf{Hash}(\hat{g}, \mathbf{ovk}_j^*)$. Since by assumption the challenge query is valid (and non-aborting), the challenger always defined this output earlier (since $\mathcal{H}_1$), and thus this intermediate tuple is already defined. Therefore, the challenger uses this newly distributed tuple $(G_j^*, H_j^*, (F_{i,j}^*)_{i=1}^{\ell})$ as the support to introducing more randomness in the computation of $\mathbf{c}_j^*$ as in $\mathcal{H}_{14}$. Namely, the challenger computes $\mathbf{c}_j^* = (d_1^{j,*}, d_2^{j,*}, (c_i^{j,*})_{i=1}^{\ell})$ as in Equation 11. Finally, the remaining part of the computation of CT_j^* is made as since $\mathcal{H}_{14}$ with the adaptation and randomization of $\boldsymbol{\sigma}_{\mathsf{trace}}^{j,b}$ as done in Equation 12 so that $\boldsymbol{\sigma}_{\mathsf{trace}}^{j,*}$ and $\boldsymbol{\pi}_{\mathsf{valid}}^{j,*}$ are computed exactly as in these previous games. So, the only change made in this game to answer to $(\mathsf{CT}_j^0, \mathsf{CT}_j^1)$ is the use of the new pre-computed intermediate tuples $(G_j^*, H_j^*, (F_{i,j}^*)_{i=1}^{\ell})$.

Next, we show that $|P_{16} - P_{15}| \leq \ell \cdot \varepsilon_{\mathsf{sxdh}} + 3\ell/p$. We only focus on the ROM variant since the simpler case of the standard model construction can be adapted in an easy way. To see why this bound holds, we first observe that $\mathcal{H}_{15}$ and $\mathcal{H}_{16}$ only differ in the way the intermediate tuples $(G_j^*, H_j^*, (F_{i,j}^*)_{i=1}^{\ell})$ are defined. Indeed, in $\mathcal{H}_{15}$ it makes no difference if all the tuples $(G_j, H_j, (F_{i,j})_{i=1}^{\ell})$ were generated as $(g, h, (f_i)_{i=1}^{\ell})^w$ at the time the challenger programmed $(C, F, G) \leftarrow \mathsf{Hash}(\hat{g}, \mathbf{ovk})$ such that $(F, G) = (f, g)^w$ for a uniform $w \in \mathbb{Z}_p$ per Hash-query. Assuming that all these tuples are pre-computed and stored when answering all the Hash-queries, given $(G_j^*, H_j^*, (F_{i,j}^*)_{i=1}^{\ell})$ both games computes CT_j^* exactly in the same manner. With this in mind, we now build $\ell + 1$ hybrid games $\mathcal{H}_{16.k}$, for $k = 0$ to ℓ, such that $\mathcal{H}_{15} = \mathcal{H}_{16.0}$ and $\mathcal{H}_{16} = \mathcal{H}_{16.\ell}$:

Game 16.k: The public key PK is defined as in $\mathcal{H}_{15}$ without secret key but with $\chi, \alpha_1, \ldots, \alpha_\ell$ such that $h = g^\chi$ and $f_1 = g^{\alpha_1}, \ldots, f_\ell = g^{\alpha_\ell}$. The challenger still uses the trapdoor extracting key ξ to compute the plaintext $\mathbf{m} = (m_i)_{i=1}^\ell$ as in the last modification brought to the way to answer to the decryption queries, i.e. from $\mathcal{H}_{12}$ to $\mathcal{H}_{16}$.

For each Hash-query on input $(\hat{g}, \mathsf{ovk})$ for a fresh $\mathsf{ovk} \in \hat{\mathbb{G}}^3$, the challenger proceeds as follows: it picks $x, y, w \leftarrow\$ \mathbb{Z}_p$,

- It computes $C = g^x h^y$ as from $\mathcal{H}_7$ to $\mathcal{H}_{16}$.
- If computes $G = g^w$ and $H = h^w$.
- For all $i \in [\ell]$ such that $i \leq k$: it picks $F_i \leftarrow\$ \mathbb{G}$.
- For all $i \in [\ell]$ such that $i > k$: it computes $F_i = f_i^w$.
- It stores the intermediate tuple $(G, H, (F_i)_{i=1}^\ell)$, and programs $(C, F, G) \leftarrow \mathsf{Hash}(\hat{g}, \mathsf{ovk})$, where $F = \prod_{i=}^\ell F_i$.

All the decryption queries are answered as from $\mathcal{H}_{12}$ to $\mathcal{H}_{16}$, and all the challenge queries $(\mathsf{CT}_j^0, \mathsf{CT}_j^1)$ are answered as in $\mathcal{H}_{15}$ and $\mathcal{H}_{16}$ from the intermediate tuple $(G_j, H_j, (F_{i,j})_{i=1}^\ell)$ that was stored for ovk_j^* in the corresponding Hash-query by emulating the randomization of CT_j^b, where $b \leftarrow\$ \{0,1\}$ was chosen at the outset of the game as usual. Moreover, the challenger still aborts and outputs a random bits if a collision on the C components occurs as from $\mathcal{H}_3$.

Now, we show that $|P_{16.k} - P_{16.k-1}| \leq \varepsilon_{\mathtt{sxdh}} + 3/p$, for all $k \in [\ell]$, where $P_{16.k} = \Pr[\mathcal{H}_{16.k} = 1]$ is the probability that the adversary $\mathcal{A}$ correctly guesses the bit b, for all $0 \leq k \leq \ell$. Let (g, f_0, G_0, F_0) be an SXDH instance. Next, we build a distinguisher $\mathcal{B}$ that uses the TCCA adversary $\mathcal{A}$ as a subroutine. First $\mathcal{B}$ picks $b \leftarrow\$ \{0,1\}$ and generates PK as in $\mathcal{H}_{15}$ and $\mathcal{H}_{16}$ except in the way it generates $(g, h, (f_i)_{i=1}^\ell)$. $\mathcal{B}$ picks $\chi \leftarrow\$ \mathbb{Z}_p$ and computes $h = g^\chi$. For all $i \in [\ell] \setminus \{k\}$, $\mathcal{B}$ picks $\alpha_i \leftarrow\$ \mathbb{Z}_p$ and computes $f_i = g^{\alpha_i}$, and sets $f_k = f_0$ as well as $f = \prod_{i=1}^\ell f_i$. The other parameters are generated as in $\mathcal{H}_{15}$ and $\mathcal{H}_{16}$, where crs_σ is in the hiding mode and crs_π is in the binding mode with a trapdoor extracting key ξ.

The distinguisher $\mathcal{B}$ programs the random oracle to answer to the q_{ovk} Hash-queries on input $(\hat{g}, \mathsf{ovk})$ with a fresh $\mathsf{ovk} \in \hat{\mathbb{G}}^3$ as follows: it picks $x, y, u, v \leftarrow\$ \mathbb{Z}_p$, and then

- It computes $C = g^x h^y$ as in $\mathcal{H}_7$-$\mathcal{H}_{16}$ and aborts as in $\mathcal{H}_3$ if a collision occurs.
- If computes $G = g^u G_0^v$, $H = G^\chi$ and $F_k = f_0^u F_0^v$.
- For all $i \in [\ell]$ such that $i < k$: it picks $F_i \leftarrow\$ \mathbb{G}$.
- For all $i \in [\ell]$ such that $i > k$: it computes $F_i = G^{\alpha_i}$.
- It stores the intermediate tuple $(G, H, (F_i)_{i=1}^\ell)$, and programs $(C, F, G) \leftarrow \mathsf{Hash}(\hat{g}, \mathsf{ovk})$, where $F = \prod_{i=}^\ell F_i$.

All the decryption queries are answered as from $\mathcal{H}_{12}$ to $\mathcal{H}_{16}$ without SK but with ξ, and all the challenge queries $(\mathsf{CT}_j^0, \mathsf{CT}_j^1)$ are answered as in $\mathcal{H}_{15}$ and $\mathcal{H}_{16}$ from the intermediate tuples $(G_j, H_j, (F_{i,j})_{i=1}^\ell)$ that was stored for ovk_j^* in the corresponding Hash-query by emulating the randomization of CT_j^b, for all $j \in [q]$. At the end of the game the adversary $\mathcal{A}$ outputs b', and $\mathcal{B}$ defines and outputs its guess as 1 if $b' = b$, and 0 otherwise.

Now, we analyze the reduction. Assuming that $g, f_0 \neq 1_{\mathbb{G}}$, let $(G_0, F_0) = (g, f_0)^\kappa (1, g^\upsilon)^\beta$ for some unknown random $\kappa, \upsilon \in \mathbb{Z}_p$ and $\beta \in \{0,1\}$. Then, the public-key elements $(g, h, (f_i)_{i=1}^\ell)$ are distributed exactly as in $\mathcal{H}_{16.k}$ and $\mathcal{H}_{16.k-1}$ since we simply have $f_k = f_0 = g^{\alpha_k}$ for some unknown random $\alpha_k \in \mathbb{Z}_p$. Therefore, $f_i = g^{\alpha_i}$ for all $i \in [\ell]$ for uniformly distributed exponents α_i. Moreover, $f = \prod_{i=1}^\ell f_i$ as in both games. We now turn to analyzing the distribution of the answer (C, F, G) to the Hash-queries on input $(\hat{g}, \mathsf{ovk})$ with a fresh $\mathsf{ovk} \in \hat{\mathbb{G}}^3$. Clearly C is independent of k and is as from $\mathcal{H}_7$. If C is involved in a challenge query, $\mathcal{B}$ responds in the same way as in both games by using the random coin (x, y) as part of the witness to compute the corresponding validity proof as from $\mathcal{H}_8$. So, we focus on (F, G). By construction, $(F_k, G) = (f_k, g)^{u + \kappa v} \cdot (g^{\upsilon v}, 1)^\beta$, where $g^{\upsilon v}$ is a random element independent to the remaining view even given $w = u + \kappa v$ conditioned on $\upsilon \neq 0$ (because v is fresh for any fresh ovk). Since $H = G^\chi = h^w$ and for all $i \in [\ell]$ we have $F_i \leftarrow\$ \mathbb{G}$ if $i < k$, and $F_i = G^{\alpha_i} = f_i^w$ if $i > k$, the intermediate tuple $(G, H, (F_i)_{i=1}^\ell)$ is distributed as in $\mathcal{H}_{16.k-1}$ if $\beta = 0$ (because $F_k = f_k^w$) and as in $\mathcal{H}_{16.k}$ if $\beta = 1$ (because $F_k = f_k^w g^{\upsilon v}$ is uniform). In both cases we also have $F = \prod_{i=1}^\ell F_i$, therefore the output (C, F, G) satisfies the distribution of $\mathcal{H}_{16.k-1}$ or $\mathcal{H}_{16.k}$ accordingly. Moreover, the decryption queries are all answered in the same way. Even if from $\mathcal{H}_{16.1}$ the CPA part $\mathbf{c}_j^*$ of the answer to a valid non-aborting challenge query $(\mathsf{CT}_j^0, \mathsf{CT}_j^1)$ are no more a randomization of $\mathbf{c}_j^b$ since the computation from Equation 11 modifies the underlying plaintext and is no more a bit-string with overwhelming probability, the simulated validity proof $\pi_{\mathsf{valid}}^{j,*}$ made for a false statement does not help the adversary $\mathcal{A}$ in producing such kind of fake proof. Indeed, since the introduction of the failure event in $\mathcal{H}_{11}$, $\mathcal{A}$ remains unable to produce a non-perfectly sound proof π_{valid} in any valid non-aborting decryption query CT as argued from the perfectly hiding nature of the opening of C known by the challenger (here, the distinguisher). This argument still holds here, and if the adversary was able to force $\mathcal{B}$ to abort for that reason with a significant difference in probability, this distinction is implicitly but automatically transmitted to the distinguishing probability here. This ends the analysis since decryption queries are computed in the same way in both games and that the challenge queries are also answered in the same way in both games since, if they are valid and non-aborting, they rely on the intermediate tuples whose distributions have already been settled.

Conclusion: To conclude, we argue that the random bit b is statistically hidden in $\mathcal{H}_{16}$. First, we note that there is no secret key SK, and that the public-key components h and $(f_i)_{i=1}^{\ell} = (g^{\alpha_i})_{i=1}^{\ell}$ are generate as h^{χ} and $f_1 = g^{\alpha_1}, \ldots, f_\ell = g^{\alpha_\ell}$, for random scalars $\chi, \alpha_1, \ldots, \alpha_\ell \hookleftarrow\$ \mathbb{Z}_p$. Let also write $(g_i)_{i=1}^{\ell} = (g^{\delta_i})_{i=1}^{\ell}$, for some random $\delta_1, \ldots, \delta_\ell \in \mathbb{Z}_p$. With probability less than ℓ/p, none of these scalars are equal to 0. Second, for each Hash-query $(\hat{g}, \mathbf{ovk})$ that generates the intermediate tuple and defines the output (C, G, F) let us write it as $(G, H, (F_i)_{i=1}^{\ell}) = (g^w, h^w, (g_i^{z_i} f_i^w))$ for random $w, z_1, \ldots, z_\ell \in \mathbb{Z}_p$. Then, $F = \prod_{i=1}^{\ell} F_i = g^{\sum_{i\in[\ell]} \delta_i z_i} f^w$. Third, the CPA part $\mathbf{c}_j^*$ of the answer CT_j^* to the valid and non-aborting challenge query $(\mathsf{CT}_j^0, \mathsf{CT}_j^1)$ can be written as

$$\begin{aligned}\mathbf{c}_j^* &= \big(d_1^{j,b}, d_2^{j,b}, (c_i^{j,b})_{i=1}^{\ell}\big) \cdot \big(g, h, (f_i)_{i=1}^{\ell}\big)^{\theta'_j} \cdot \big(G_j^*, H_j^*, (F_{i,j}^*)_{i=1}^{\ell}\big)^{\theta''_j} \\ &= \big(d_1^{j,b}, d_2^{j,b}, (c_i^{j,b})_{i=1}^{\ell}\big) \cdot \big(g, h, (f_i)_{i=1}^{\ell}\big)^{\theta'_j + w_j^* \theta''_j} \cdot \big(1, 1, (g_i^{z_{i,j}^* \theta''_j})_{i=1}^{\ell}\big).\end{aligned}$$

for the random coins $w_j^*, z_{1,j}^*, \ldots, z_{\ell,j}^* \in \mathbb{Z}_p$ defined previously in the Hash-query $(\hat{g}, \mathbf{ovk}_j^*)$ and that can be used a single time since the trace $\mathbf{ovk}_j^*$ cannot be repeated through different challenge queries. Hence, given $(G_j^*, H_j^*) = (g, h)^{w_j^*}$ and $F_j^* = g^{\sum_{i\in[\ell]} \delta_i z_{i,j}^*} f^{w_j^*}$, the computation of $\mathbf{c}_j^*$ still perfectly hides $\mathbf{c}_j^b$. Indeed, w_j^* and $\sum_{i\in[\ell]} \delta_i z_{i,j}^*$ makes G_j^* and F_j^* uniform in $\mathbb{G}$ and $H_j^* = (G_j^*)^{\chi}$. Moreover, $\theta'_j + w_j^* \theta''_j$ perfectly randomizes $(d_1^{j,b}, d_2^{j,b})$ with the component (g, h) which makes it independent of b. That is because all valid non-aborting challenge queries $(\mathsf{CT}_j^0, \mathsf{CT}_j^1)$ are also computed with perfectly sound proofs that their respective CPA part $(\mathbf{c}_j^0, \mathbf{c}_j^1)$ are honest due to the change introduced in $\mathcal{H}_{11}$. Then, $(d_1^{j,0}, d_2^{j,0}), (d_1^{j,1}, d_2^{j,1}) \in \mathrm{span}\langle(g, h)\rangle$. Furthermore, all the $c_i^{j,b}$ components are also perfectly hidden by the corresponding $z_{i,j}$ for $i = 1$ to $\ell - 1$. Since F_j^* reveals one linear equation in all the $z_{i,j}$, we cannot directly argue about the uniformity of the last component $c_\ell^{j,b}$. Nevertheless, it suffices to see that θ''_j hides $c_\ell^{j,b}$ with $g_\ell^{z_{\ell,j}^*}$ as long as $g_\ell \neq 1_{\mathbb{G}}$ and $z_{\ell,j}^* \neq 0$, for which the latter condition always holds but with probability $1/p$. Fourth, the validity proof $\pi_{\mathsf{valid}}^{j,*}$ are simulated independently of b since the proof is given the above $\mathbf{c}_j^*$ and uses the witness (x_j^*, y_j^*) of $C_j^* = g^{x_j^*} h^{y_j^*}$ and the one in $(A, B) = (g, h)^z$ from PK. Fifth, the tracing part $\boldsymbol{\sigma}_{\mathsf{trace}}^{j,*} = (\boldsymbol{C}_R^{j,*}, \boldsymbol{C}_S^{j,*}, \boldsymbol{\sigma}_2^{j,*}, \boldsymbol{\sigma}_3^{j,*}, \hat{\boldsymbol{\pi}}_{sig}^{j,*})$ is computed such that $\boldsymbol{\sigma}_2^{j,*} = \boldsymbol{\sigma}_2^{j,0} = \boldsymbol{\sigma}_2^{j,1}$ and $\boldsymbol{\sigma}_3^{j,*} = \boldsymbol{\sigma}_3^{j,0} = \boldsymbol{\sigma}_3^{j,1}$ since CT_j^* is computed only for non-aborting query as defined from $\mathcal{H}_2$. Also, the Groth-Sahai proof $(\boldsymbol{C}_R^{j,*}, \boldsymbol{C}_S^{j,*}, \hat{\boldsymbol{\pi}}_{sig}^{j,*})$ is a valid proof that there exists a valid LHSP signature for $\mathbf{ovk}_j^*$ on the vector $(g, \prod_{i=1}^{\ell} c_i^{j,*}, d^{j,*})$ composed from $\mathbf{c}_j^*$. Since $\mathbf{crs}_\sigma$ is in the hiding mode, the perfect witness indistinguishability thus reveals nothing about how the underlying signatures has been performed, and then remains independent of $\boldsymbol{\sigma}_{\mathsf{trace}}^{j,b}$ (because the ability of computing such a proof was also independent of b). Consequently, $P_{16} \leq 1/2 + (\ell + q)/p$ for the ROM-based construction, and

$P_{16} \leq 1/2 + (\ell+1)/p$ for the standard model construction. For the ROM-based construction, we find

$$\varepsilon_{\mathtt{tcca}} \leq \frac{1}{2} + (\ell+6)\varepsilon_{\mathtt{sxdh}} + \frac{4q + q_{\mathsf{ovk}}(q_{\mathsf{ovk}}-1)/2 + q_{\mathsf{dec}} + 4\ell + 8}{p},$$

where q is the number of challenge queries, q_{dec} is the number of decryption queries, and q_{ovk} is the number of Hash-queries of the form $(\hat{g}, \mathbf{ovk})$ made by the adversary. For the standard model construction, we have $\varepsilon_{\mathtt{tcca}} \leq \frac{1}{2} + (\ell+6)\varepsilon_{\mathtt{sxdh}} + \varepsilon_{\mathtt{cr}} + (q_{\mathsf{dec}} + 4\ell + 9)/p$. $\square$

4 TREnc for Mixnet

In some cases, it is not desirable to encrypt a bit-by-bit decomposition of the message but rather preferable to encode many bits in a group element, such as in [14]. Indeed, in some voting protocols, the ballots are decrypted after passing through a mixing network and the validity of the decrypted vote can be verified directly. Hence we propose a modification of our construction that can be used efficiently with mixing networks.

4.1 Intuition

First, instead of encrypting the message bit-by-bit in the CPA encryption part, it is tempting to encrypt directly $c_i = m_i f_i^\theta$ for each group element m_i. Unfortunately, the different c_i cannot be compressed anymore in the tracing part, as the compression is not a binding commitment of the messages anymore. There is thus little advantage to batch several group elements in one ciphertext and one could run the encryption mechanism for a single element on each entry. Furthermore, it is possible in practice to pack a few hundred of bits in one group element. Hence we focus on the case $l = 1$ in our construction.

Most of the intuition given for the bit-by-bit scheme still holds. The main difference is that we do not need to prove most of the relations from equations (6) and (7) anymore as it is not necessary to prove the encryptions of bits. It remains essentially to construct a ranomizable simulation-sound proof that $(d_1, d_2) \in \langle (g, h) \rangle$. We could use the same approach as the construction using the GS-based OR-techniques from [23], but using two CRSes and simulating the second branch for the π_1 proof is costly, as it requires 5 elements of $\mathbb{G}$ and 10 elements of $\hat{\mathbb{G}}$.

Instead, we are using a new technique that uses an additional pair of commitments (Y_1, Y_2) that are normally redundant. More precisely, the proof consist in $(X_1, X_2) = (g, h)^a$ and $(Y_1, Y_2) = (g, h)^b$ which are the commitments of some values (a, b), a value E which is derived from $\mathsf{Hash}(\mathsf{ovk})$, and a response $\hat{Z} = E^\theta \hat{g}^a \hat{h}^b$. The values verify the pairing equations:

$$e(d_1, E) \cdot e(X_1, \hat{g}) \cdot e(Y_1, \hat{h}) = e(g, \hat{Z})$$
$$e(d_2, E) \cdot e(X_2, \hat{g}) \cdot e(Y_2, \hat{h}) = e(h, \hat{Z})$$

We can simulate this proof by using a trapdoor in E and h. Namely, we generate E as a Pedersen commitment $\hat{g}^x \hat{h}^y$ and h as g^ω. Then, for a random $\hat{Z} = \hat{g}^s \hat{g}^t$,

$$X_1 = g^s / d_1^\alpha \quad Y_1 = h^t / d_1^\beta$$
$$X_2 = g^s / d_2^\alpha \quad Y_2 = h^t / d_1^\beta$$

form a valid simulated proof. As E is linked to a given ovk, we separate the simulated proofs for each different trace, similarly to C in the previous construction.

Still, the adversary cannot simulate this proof. If it manages to find a proof $(X_1, X_2, Y_1, Y_2, \hat{Z})$ in a pre-challenge query for a statement $(d_1, d_2) \notin \langle (g, h) \rangle$, then we have:

$$e(d_1^\omega / d_2, E) \cdot e(X_1^\omega / X_2, \hat{g}) \cdot e(Y_1^\omega / Y_2, \hat{h}) = e(g^\omega / h, Z) = 1$$

for a non-trivial $D = d_1^\omega / d_2$. Also, at least one of $X = X_1^\omega / X_2$ or $Y = Y_1^\omega / Y_2$ is non trivial and we can rewrite the previous equation as

$$e(D, \hat{g}^x \hat{h}^y) \cdot e(X, \hat{g}) \cdot e(Y, \hat{h}) = e(XD^x, \hat{g}) \cdot e(YD^y, \hat{h}) = 1.$$

Since the x is perfectly hidden from the adversary when he chooses the proof, XD^x will be non-trivial with overwhelming probability and we can reduce to SXDH.

Unfortunately, this technique would not be competitive in the bit-by-bit case. Indeed, to take advantage of the perfectly hidding property of the Pedersen commitment E, we need two commitments X_i and Y_i for each relation that needs to be proven. The cost is thus asymptotically twice as big as the previous construction and it is thus more efficient to pay the fix-cost of the OR-branches technique that is cheaper in the long term.

As the previous construction, we simultaneously define a construction in the standard model and a variant in the ROM.

4.2 Description

Gen(1^λ): Run $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$.

1. Generate $h, E_0, E_1, F, G \leftarrow\!\!\$\, \mathbb{G}^5, \hat{h} \leftarrow\!\!\$\, \hat{\mathbb{G}}$. In the ROM construction, ignore E_0, E_1, F, G and generate all the other values from $\mathsf{Hash}(g, \hat{g}, \ell)$ instead, where $g, \hat{g} \in \mathsf{pp}$.
2. Pick a random pair $\mathsf{sk} = (\alpha, \beta) \leftarrow\!\!\$\, \mathbb{Z}_p^2$ of scalars and compute $f = g^\alpha h^\beta$.
3. Generate a tuple of 4 random group elements $\mathbf{crs}_\sigma \leftarrow\!\!\$\, \mathbb{G}^4$ is seen as a Groth-Sahai CRS to commit to group elements over $\mathbb{G}$. For simplicity, we write Com for the commitment algorithm $\mathsf{Com}(\mathbf{crs}_\sigma, \cdot)$.
   In the ROM construction, generate this CRS from $\mathsf{Hash}(g, \hat{g}, f)$ instead.

The private key consists of $\mathsf{SK} = \mathsf{sk} = (\alpha, \beta)$ and the public key $\mathsf{PK} \in \mathbb{G}^{11} \times \hat{\mathbb{G}}^2$ that can be derived from pp and f only is

$$\mathsf{PK} = \left(g, h, f, E_0, E_1, F, G, \mathbf{crs}_\sigma, \hat{g}, \hat{h}\right).$$

In the ROM construction, the public key can be derived from pp and f only and the values E_0, E_1, F, G are stripped from PK.

Enc(PK, $\boldsymbol{m}$): Given a message $m \in \mathbb{G}$ to encrypt, first run $\underline{\mathsf{LGen}(\mathsf{PK})}$: Generate a key pair $(\mathbf{osk}, \mathbf{ovk})$ for the one-time linearly homomorphic signature of Section 2.3 from the public generators $\hat{g}, \hat{h}$ in order to sign vectors of dimension 3. Let $\mathsf{lk} = \mathbf{osk} = \{(\eta_i, \zeta_i)\}_{i=1}^3$ be the private key, of which the corresponding public key is $\mathbf{ovk} = \{\hat{l}_i\}_{i=1}^3$. From $\mathbf{ovk}$, compute $\tau = \mathsf{Hash}(\hat{g}, \mathbf{ovk})$ and $C = E_0^\tau E_1$. In the ROM construction, derive $(C, F, G) = \mathsf{Hash}(\hat{g}, \mathbf{ovk})$.

Then, conduct the following steps of $\underline{\mathsf{LEnc}(\mathsf{PK}, \mathsf{lk}, \boldsymbol{m})}$:

1. Pick $\theta \leftarrow\!\!\$\, \mathbb{Z}_p$ and compute the CPA encryption $\mathbf{c} = (d_1, d_2, d_3)$, where $d_1 = g^\theta$, $d_2 = h^\theta$ and $d_3 = mf^\theta$. Keep the random coin θ.
   Next steps 2-3 are dedicated to the tracing part $\boldsymbol{\sigma}_{\mathsf{trace}}$.
2. To allow tracing, use $\mathsf{lk} = \mathbf{osk}$ to authenticate the row space of the matrix $\mathbf{T} = \left(T_{i,j}\right)_{1 \le i,j \le 3}$

$$\mathbf{T} = \begin{pmatrix} g & d_3 & d_1 \\ 1 & f & g \\ 1 & F & G \end{pmatrix}, \tag{13}$$

   Namely, sign each row $\boldsymbol{T}_i$ of $\mathbf{T}$ resulting in $\boldsymbol{\sigma} = (\boldsymbol{\sigma}_i)_{i=1}^3$, where $\boldsymbol{\sigma}_i = (R_i, S_i) \in \mathbb{G}^2$.
3. Commit to $\boldsymbol{\sigma}_1 = (R_1, S_1)$ using $\mathbf{crs}_\sigma$ as $\boldsymbol{C}_R = \mathsf{Com}(R_1)$, $\boldsymbol{C}_S = \mathsf{Com}(S_1)$. To ensure that each $\boldsymbol{\sigma}_1$ is a valid one-time LHSP signature on (g, d_3, d_1), compute the Groth-Sahai proof $\hat{\boldsymbol{\pi}}_{sig} \in \hat{\mathbb{G}}^2$ that

$$\begin{aligned} & e(R_1, \hat{g}) \cdot e(S_1, \hat{h}) \\ & = e(g, \hat{l}_1) \cdot e(d_3, \hat{l}_2) \cdot e(d_1, \hat{l}_3) \end{aligned}$$

   The next step shows the validity of $\mathbf{c}$ *associated to* ovk.

4. Given θ, compute the randomizable simulation-sound proof π_0 that $(d_1, d_2) \in \langle g, h \rangle$. More precisely, compute the commitments $(X_1, X_2) = (g, h)^a$ and $(Y_1, Y_2) = (g, h)^b$ for random $a, b \in \mathbb{Z}_p$.
   Then, compute the response $\hat{Z} = E^\theta \hat{g}^a \hat{h}^b$ that verifies the equations:
$$e(d_1, E) \cdot e(X_1, \hat{g}) \cdot e(Y_1, \hat{h}) = e(g, \hat{Z})$$
$$e(d_2, E) \cdot e(X_2, \hat{g}) \cdot e(Y_2, \hat{h}) = e(h, \hat{Z})$$
   Let $\boldsymbol{\pi}_{\mathsf{valid}} = (X_1, X_2, Y_1, Y_2, \hat{Z}) \in \mathbb{G}^4 \times \hat{\mathbb{G}}$

Output the ciphertext
$$\mathsf{CT} = \Big(\mathbf{c}, \mathbf{ovk}, \boldsymbol{\sigma}_{\mathsf{trace}}, \boldsymbol{\pi}_{\mathsf{valid}}\Big),$$
where $\boldsymbol{\sigma}_{\mathsf{trace}} = (\boldsymbol{C}_R, \boldsymbol{C}_S, \sigma_2, \sigma_3, \hat{\boldsymbol{\pi}}_{sig}) \in \mathbb{G}^8 \times \hat{\mathbb{G}}^2$ and $\boldsymbol{\pi}_{\mathsf{valid}} \in \mathbb{G}^4 \times \hat{\mathbb{G}}$.

Trace(PK, CT): Parse PK and CT as above, and output **ovk**.

Rand(PK, CT): Parse the ciphertext as above, and compute (C, F, G) from $\mathsf{Hash}(\hat{g}, \mathbf{ovk})$ and PK as above and conduct the following steps:

1. Randomize the CPA part $\mathbf{c} = (d_1, d_2, d_3)$: Pick random $\theta' \leftarrow\!\$\, \mathbb{Z}_p$ and compute $\mathbf{c}' = (d_1', d_2', d_3') = (d_1 \cdot g^{\theta'}, d_2 \cdot h^{\theta'}, d_3 \cdot f^{\theta'}) = \mathbf{c} \cdot (g, h, f)^{\theta'}$.
   Next steps 2-3 are dedicated to the tracing part $\boldsymbol{\sigma}'_{\mathsf{trace}}$.
2. Adapt the commitments $\boldsymbol{C}_R$ and $\boldsymbol{C}_S$ to the signature $\boldsymbol{\sigma}_1$ so that it it becomes commitments $\tilde{\boldsymbol{C}}_R$ and $\tilde{\boldsymbol{C}}_S$ to the signature $\boldsymbol{\sigma}_1' = \boldsymbol{\sigma}_1 \boldsymbol{\sigma}_2^{\theta'}$. That is, parse $\boldsymbol{\sigma}_2$ as (R_2, S_2) and compute $\tilde{\boldsymbol{C}}_R = \boldsymbol{C}_R \cdot \iota(R_2^{\theta'})$ and $\tilde{\boldsymbol{C}}_S = \boldsymbol{C}_S \cdot \iota(S_2^{\theta'})$. Note: $\boldsymbol{\sigma}_1'$ is a valid signature on (g, d_3', d_1') for **opk**, and $\hat{\boldsymbol{\pi}}_{sig}$ remains a proof that $\boldsymbol{\sigma}_1' = (R_1', S_1')$ satisfies
$$\begin{aligned} & e(R_1', \hat{g}) \cdot e(S_1', \hat{h}) \\ & = e(g, \hat{l}_1) \cdot e(d_3, \hat{l}_2) \cdot e(d_1, \hat{l}_3) \end{aligned}$$
   where $(R_1', S_1') = (R_1, S_1) \cdot (R_2, S_2)^{\theta'}$.
3. Randomize the GS proof $(\tilde{\boldsymbol{C}}_R, \tilde{\boldsymbol{C}}_S, \hat{\boldsymbol{\pi}}_{sig})$ for the CRS $\mathbf{crs}_\sigma$ leading to $(\boldsymbol{C}'_R, \boldsymbol{C}'_S, \hat{\boldsymbol{\pi}}'_{sig})$. Let the randomized tracing part be $\boldsymbol{\sigma}'_{\mathsf{trace}} = (\boldsymbol{C}'_R, \boldsymbol{C}'_S, \sigma_2, \sigma_3, \hat{\boldsymbol{\pi}}'_{sig})$[7].
   The next step is dedicated to the validity proof $\boldsymbol{\pi}'_{\mathsf{valid}}$.
4. Adapt the proof $\boldsymbol{\pi}_{\mathsf{valid}} = (X_1, X_2, Y_1, Y_2, Z)$ to (d_1', d_2') using θ'. Namely, compute $(X_1, X_2)' = (X_1, X_2) \cdot (g, h)^{a'}$, $(Y_1, Y_2)' = (Y_1, Y_2) \cdot (g, h)^{b'}$ and $\hat{Z}' = \hat{Z} \cdot E^{\theta'} \hat{g}^{a'} \hat{h}^{b'}$ for random $a', b' \in \mathbb{Z}_p$. Let $\boldsymbol{\pi}'_{\mathsf{valid}} = (X_1', X_2', Y_1', Y_2', \hat{Z}')$

Return the randomized ciphertext
$$\mathsf{CT}' = \Big(\mathbf{c}', \mathbf{ovk}, \boldsymbol{\sigma}'_{\mathsf{trace}}, \boldsymbol{\pi}'_{\mathsf{valid}}\Big).$$

Ver(PK, CT): Conduct the following checks:

1. Verify whether PK and CT parse properly. If not, output 0. Else compute (C, F, G) from $\mathsf{Hash}(\hat{g}, \mathbf{ovk})$ and PK as above.
   Next steps 2-3 verify the tracing part $\boldsymbol{\sigma}_{\mathsf{trace}}$.
2. Verify the validity of the signatures σ_2 and σ_3 on the last two rows $\boldsymbol{T}_1, \boldsymbol{T}_2$ of the matrix $\mathbf{T}$ in (13) with respect to $\mathbf{ovk} = \{\hat{l}_i\}_{i=1}^3$. Namely, parse $\boldsymbol{\sigma}_i = (R_i, S_i)$ for $i = 2, 3$ and check the next equations:
$$e(R_2, \hat{g}) e(S_2, \hat{h}) = e(f, \hat{l}_2) \cdot e(g, \hat{l}_3)$$
$$e(R_3, \hat{g}) e(S_3, \hat{h}) = e(F, \hat{l}_2) \cdot e(G, \hat{l}_3)$$

[7] Since computing these LHSP signatures is deterministic in $\mathsf{lk} = \mathbf{osk}$, they cannot be randomized, and we do not need to randomize them to satisfy the strong randomizability notion.

3. Verify the validity of the proof $\hat{\boldsymbol{\pi}}_{sig}$ that committed variables in $(\boldsymbol{C}_R, \boldsymbol{C}_S)$ consist of a signature on the first row (g, d_3, d_1) of $\mathbf{T}$ under **ovk**.
   Next step verifies the validity proof part $\boldsymbol{\pi}_{\mathsf{valid}}$.
4. Verify the proof $\boldsymbol{\pi}_{\mathsf{valid}}$. Namely, parse the proof $\boldsymbol{\pi}_{\mathsf{valid}} = (X_1, X_2, Y_1, Y_2, Z)$, and check

$$e(d_1, E) \cdot e(X_1, \hat{g}) \cdot e(Y_1, \hat{h}) = e(g, Z)$$
$$e(d_2, E) \cdot e(X_2, \hat{g}) \cdot e(Y_2, \hat{h}) = e(h, Z)$$

Output 1 if all these checks pass, otherwise, output 0.

Dec(SK, PK, CT): If $\mathsf{ver}(\mathsf{PK}, \mathsf{CT}) = 0$, output $\perp$. Otherwise, given $\mathsf{SK} = (\alpha, \beta)$ and $\mathbf{c} = (d_1, d_2, d_3)$ included in CT, compute and output $m = d_3 \cdot d_1^{-\alpha} d_2^{-\beta}$.

The ciphertext consists of 15 group elements in $\mathbb{G}$ and 6 group elements in $\hat{\mathbb{G}}$.

5 Performance & comparison

5.1 Performance & comparison

We provide a Rust implementation [3] to evaluate the time to encrypt and craft ballots encrypting votes of various sizes and styles using the construction of BeleniosRF [10] for the encryption of bits, the original TREnc construction for the encryption of group elements [14] and our bit-by-bit and group-encoded TREncs. We assume that the critical computation steps are done by the voter client-side, as they could typically use a smartphone, a browser or a low-end voting device. On the other hand, the operations on the rerandomization server can be done on the fly as ballots come in.

We conducted our experiments on a laptop with a 1.8Ghz Intel i7 processor and 16GB of RAM running Ubuntu 20.04 LTS. We used the bls12_381 Rust crate [2] with the pairing-friendly elliptic curve of the same name and the criterion library to perform the benchmarks. For our comparison with BeleniosRF, we reimplemented in Rust the referenced Javascript implementation [1].

Table 1: Time to encrypt the vote of a ballot containing $\ell = 1, 2, 4, 8, 64$ bits or group element. The times reported are in milliseconds.

Bit-by-bit constructions							
Number of bits ℓ	1	2	4	8	64	Group elements	Exponentiations
BeleniosRF [10]	58.5	82.2	124.7	191.0	1211.7	$\mathbb{G}^{6\ell+14} \times \hat{\mathbb{G}}^{6\ell+7}$	$\mathbb{G}^{9\ell+23} \times \hat{\mathbb{G}}^{8\ell+17}$
This paper	67.3	78.4	95.1	132.6	628.7	$\mathbb{G}^{4\ell+13} \times \hat{\mathbb{G}}^{4\ell+15}$	$\mathbb{G}^{4\ell+33} \times \hat{\mathbb{G}}^{4\ell+26}$
Group-encoded constructions							
Original TREnc [14]	27.3					$\mathbb{G}^{13} \times \hat{\mathbb{G}}^{5}$	$\mathbb{G}^{29} \times \hat{\mathbb{G}}^{10}$
This paper	32.6					$\mathbb{G}^{15} \times \hat{\mathbb{G}}^{6}$	$\mathbb{G}^{25} \times \hat{\mathbb{G}}^{13}$

We see in Table 1 that even for a 64-choices race, the encrypting time of the bit-by-bit TREnc is beneath one second. For the variant of the TREnc that encodes messages as a group element of $\mathbb{G}$, the encrypting time remains constant (until messages are too big to be fitted in one element of $\mathbb{G}$, which may happen around 360 bits depending on the chosen encoding method – but one may want to avoid packing too much information in one group element anyway, in order to avoid having unique ballots).

BeleniosRF is only compatible with a short (logarithm) bit-by-bit decomposition of the message (in the security parameter) and is slightly less efficient, mostly because of the costly quadratic Groth-Sahai proofs. Moreover, this mechanism also needs another computational assumption in addition to SXDH. On the other hand, the original TREnc construction for group-encoded messages is slightly more efficient than the one we propose here, but it does not have a public coin CRS. A full comparison composed of the size of the ciphertexts and the number of exponentiations to encrypt a message is also given in Table 1.

6 Application to E-Voting

As of today, the main application of TREncs is receipt-free voting. Receipt-Freeness is a security property of voting schemes ensuring that a voter cannot craft a proof (the receipt) to convince another party of how they have voted [7]. For example, a compliant voter might be incentivized by a vote buyer to vote in a certain way in exchange of some money or under some threat. A definition of receipt-freeness for single-pass voting schemes is given in Appendix A.

It has been shown in [14] that we can generically instantiate a receipt-free protocol voting from a TREnc. Even more, the reduction uses a tight transition equivalent to our mTCCA definition to prove the receipt-freeness, but with a linear loss of security since their scheme is not tightly mTCCA secure. On the contrary, we can instantiate a voting system from our TREnc scheme that is almost tightly receipt-free.

Intuitively, this generic voting scheme makes use of an intermediate server that rerandomizes ballots before publishing them. Here, the TCCA property of the TREnc essentially guarantees the indistinguishability of the rerandomization of a ballot containing some vote to another ballot containing another vote from the same voter.

Apart from receipt-freeness, verifiability and privacy are two important notions in electronic-voting. Intuitively, a protocol is verifiable if, by performing a determined set of verifications, the participants have the guarantee that their vote is properly recorded and that the tally is properly computed from the set of recorded votes. In a TREnc-based receipt-free protocol the traceable property precisely ensures a voter that if they see a ballot on the public bulletin board with the same trace as the ballot they sent to the rerandomization server, then the content of the ballot has remained unaltered (assuming the secrecy of the link key).

Regarding privacy, we can transform the key generation algorithm of our construction into a distributed key generation (DKG) using a standard technique [17]. At the end of the protocol, all the parties that behave honestly get a secret share of $\mathsf{sk} = (\alpha_1, \ldots, \alpha_\ell)$ so that any set of $t+1$ shares allows recomputing sk by Lagrange interpolation. Each secret share sk_j comes along with an associated public key pk_j allowing party j to prove she honestly computes decryption shares through an additional (simulation-sound) ZK proof. Such proofs are useful for the talliers to ensure correctness of the election outcome. We do not discuss these standard proofs here.

Acknowledgments. Henri Devillez is supported by a FRIA grant and Thomas Peters is research associate of the Belgian Fund for Scientific Research (F.R.S.-FNRS).

References

1. Beleniosrf javascript implementation (2016), https://gist.github.com/pyrros/4fddd7d49ae7c9c935f5d6a9a27d14c3#file-belenios-booth-js-L539
2. bls12_381 rust crate (2023), https://github.com/zkcrypto/bls12_381
3. Tight trenc implementation (2024), https://https://github.com/uclcrypto/tight-trenc
4. Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K., Ohkubo, M.: Structure-preserving signatures and commitments to group elements. In: Advances in Cryptology – CRYPTO 2010. pp. 209–236. Springer (2010)
5. Abe, M., Haralambiev, K., Ohkubo, M.: Signing on elements in bilinear groups for modular protocol design. Cryptology ePrint Archive, Report 2010/133 (2010)
6. Bellare, M., Boldyreva, A., Micali, S.: Public-key encryption in a multi-user setting: Security proofs and improvements. In: Preneel, B. (ed.) Advances in Cryptology - EUROCRYPT 2000. LNCS, vol. 1807, pp. 259–274. Springer (2000)
7. Benaloh, J.C., Tuinstra, D.: Receipt-free secret-ballot elections (extended abstract). In: Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing. pp. 544–553. ACM (1994)
8. Bernhard, D., Cortier, V., Galindo, D., Pereira, O., Warinschi, B.: SoK: A Comprehensive Analysis of Game-Based Ballot Privacy Definitions. In: S&P'15. IEEE Computer Society (2015)
9. Blazy, O., Fuchsbauer, G., Pointcheval, D., Vergnaud, D.: Signatures on randomizable ciphertexts. In: Public Key Cryptography - PKC 2011. LNCS, vol. 6571, pp. 403–422. Springer (2011)
10. Chaidos, P., Cortier, V., Fuchsbauer, G., Galindo, D.: BeleniosRF: A Non-interactive Receipt-Free Electronic Voting Scheme. In: CCS'16. ACM (2016)

11. Chase, M., Kohlweiss, M., Lysyanskaya, A., Meiklejohn, S.: Malleable proof systems and applications. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 281–300. Springer (2012)
12. Couteau, G., Hartmann, D.: Shorter non-interactive zero-knowledge arguments and zaps for algebraic languages. In: Advances in Cryptology - CRYPTO 2020. LNCS, vol. 12172, pp. 768–798. Springer (2020)
13. Devillez, H., Pereira, O., Peters, T., Yang, Q.: Can we cast a ballot as intended and be receipt free? In: 2024 IEEE Symposium on Security and Privacy (SP). pp. 172–172. IEEE Computer Society (may 2024)
14. Devillez, H., Pereira, O., Peters, T.: Traceable receipt-free encryption. In: Agrawal, S., Lin, D. (eds.) Advances in Cryptology - ASIACRYPT 2022. LNCS, vol. 13793, pp. 273–303. Springer (2022)
15. Doan, T.V.T., Pereira, O., Peters, T.: Encryption mechanisms for receipt-free and perfectly private verifiable elections. In: Applied Cryptography and Network Security, ACNS 2024. LNCS, vol. 14583, pp. 257–287. Springer (2024)
16. Fuchsbauer, G.: Commuting signatures and verifiable encryption and an application to non-interactively delegatable credentials. Cryptology ePrint Archive (2010)
17. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure Distributed Key Generation for Discrete-Log Based Cryptosystems. In: EUROCRYPT 1999. Springer (1999)
18. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Advances in Cryptology – EUROCRYPT 2008. pp. 415–432. Springer (2008)
19. Hirt, M., Sako, K.: Efficient receipt-free voting based on homomorphic encryption. In: Advances in Cryptology - EUROCRYPT 2000. LNCS, vol. 1807, pp. 539–556. Springer (2000)
20. Kiltz, E., Wee, H.: Quasi-adaptive nizk for linear subspaces revisited. Cryptology ePrint Archive, Report 2015/216 (2015)
21. Libert, B., Peters, T., Joye, M., Yung, M.: Linearly homomorphic structure-preserving signatures and their applications. In: Advances in Cryptology - CRYPTO 2013. LNCS, vol. 8043, pp. 289–307. Springer (2013)
22. Pointcheval, D.: Linearly-Homomorphic Signatures for Short Randomizable Proofs of Subset Membership. In: Eighth International Joint Conference on Electronic Voting (E-Vote-ID '23). Luxembourg, Luxembourg (Oct 2023)
23. Ràfols, C.: Stretching groth-sahai: NIZK proofs of partial satisfiability. In: Theory of Cryptography, TCC. LNCS, vol. 9015, pp. 247–276. Springer (2015)
24. Shoup, V., Gennaro, R.: Securing threshold cryptosystems against chosen ciphertext attack. J. Cryptol. **15**(2), 75–96 (2002)

[Security proofs]

A Voting Scheme & Receipt-Freeness

A.1 Voting Protocol syntax

Let $\mathcal{C}$ be a set of voting options (*e.g.* candidates or ordered lists of candidates), $\mathcal{R}$ be a set of results (*e.g.* the name of the winner(s)) and $\rho : \mathcal{C}^* \to \mathcal{R}$ be some counting function. The goal of a voting protocol is to evaluate ρ on the private choices of the voters, in a verifiable manner. In our work, a *voting protocol* is a tuple of protocols $(\mathsf{Setup}, \mathsf{Vote}, \mathsf{Valid}, \mathsf{Append}, \mathsf{Publish}, \mathsf{TraceBallot}, \mathsf{VerifyVote},$
$\mathsf{Tally}, \mathsf{Verify})$ which involve the following parties:

- The *election administrator* EA organizes the election and coordinates the phases of the protocol.
- The *public board* PB is a public, append-only board where various information concerning the voting protocol are stored.
- The *voters* cast an encrypted ballot to the rerandomization server.
- The *rerandomization server* RS receives the encrypted ballots submitted by the voters, rerandomizes them and submits them to the public board.
- The *talliers* T compute the result of the election from the (encrypted) valid ballots.
- The *auditors* A check that the public board is consistent and that the result is correct with respect to the public board.

Those protocols have the following signatures and functionalities:

- Setup is a protocol run by EA and T. It takes as input λ, the security parameter and outputs some cryptographic information such as the group used and the public encryption key. They are stored in the public board. That protocol involves EA and T, so that at the end of the protocol, the talliers each have a share of the decryption key. To simplify the notations, we denote $(\mathsf{pk}, \mathsf{sk}) \longleftarrow \mathsf{Setup}(\lambda)$ the output of the setup, even if sk may never be explicitly created or if the setup may output additional information.
- Vote is an algorithm run by a voter. It takes as input a voting intention $v \in \mathcal{C}$ and outputs a ballot b which is sent to RS.
- Valid is an algorithm that takes as input a ballot b and the public board PB. It outputs 1 if the ballot is valid with respect to the public board, and 0 otherwise.
- Append is an algorithm run by RS. It that takes as input the bulletin board PB and a ballot ballot. It updates PB with a processing of ballot (e.g. a rerandomization).
- $\mathsf{Publish}$ is an algorithm that takes an input the bulletin board PB and outputs its public view.
- $\mathsf{TraceBallot}$ is an algorithm that takes as input a ballot b and outputs a tag t. The tag is the information that the voter can use to track their ballot with the $\mathsf{VerifyVote}$ algorithm.
- $\mathsf{VerifyVote}$ is an algorithm run by the voter to verify that their vote is indeed contained in PB. It takes as input the bulletin board PB and a tag t and should return true if the voter is convinced that their last ballot has been recorded correctly on PB.
- Tally is a protocol run by the talliers. It uses as input their shares of the election secret key and PB to compute the result r of the election, as well as a transcript Π which is used for verifiability.
- Verify takes as input $r \in \mathcal{R}$, PB and the transcript Π and outputs 1 of the data are consistent, 0 otherwise. This algorithm is ran by the auditors.

A.2 Receipt-freeness

For single-pass voting schemes, [10] formalized this notion into a game-based definition, which has been extended to voting schemes with tracing mechanisms [14].

Definition 6 (Receipt-Freeness). *A voting system $\mathcal{V}$ has receipt-freeness if there exists PPT algorithms* $\mathsf{SimSetupElection}$ *and* $\mathsf{SimProof}$ *such that no* PPT *adversary $\mathcal{A}$ can distinguish between games $\mathsf{Exp}^{\mathsf{srf},0}_{\mathcal{A},\mathcal{V}}(\lambda)$ and $\mathsf{Exp}^{\mathsf{srf},1}_{\mathcal{A},\mathcal{V}}(\lambda)$ defined by the oracles in Figure 2, that is for any efficient algorithm $\mathcal{A}$:*

$$\left| Pr\left[\mathsf{Exp}^{\mathsf{srf},0}_{\mathcal{A},\mathcal{V}}(\lambda) = 1\right] - Pr\left[\mathsf{Exp}^{\mathsf{srf},1}_{\mathcal{A},\mathcal{V}}(\lambda) = 1\right]\right|$$

is negligible in λ.

```
$\mathcal{O}\mathsf{init}(\lambda)$
if $\beta = 0$ then $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$
else $(\mathsf{pk}, \mathsf{sk}, \tau) \leftarrow \mathsf{SimSetupElection}(1^\lambda)$
$\mathsf{PB}_0 \leftarrow \perp; \mathsf{PB}_1 \leftarrow \perp$
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 $\mathsf{Valid}(\mathsf{PB}_0, \mathsf{b}_0) = 0$ or $\mathsf{Valid}(\mathsf{PB}_1, \mathsf{b}_1) = 0$
then return $\perp$
else $\mathsf{Append}(\mathsf{PB}_0, \mathsf{b}_0); \mathsf{Append}(\mathsf{PB}_1, \mathsf{b}_1)$
```

```
$\mathcal{O}\mathsf{board}()$
return $\mathsf{Publish}(\mathsf{PB}_\beta)$
```

```
$\mathcal{O}\mathsf{tally}()$
$(\mathsf{r}, \Pi) \leftarrow \mathsf{Tally}(\mathsf{PB}_0, \mathsf{sk})$
if $\beta = 1$ then $\Pi \leftarrow \mathsf{SimProof}(\mathsf{PB}_1, \mathsf{r})$
return $(\mathsf{r}, \Pi)$
```

Fig. 2: Oracles used in the $\mathsf{Exp}^{\mathsf{srf},\beta}_{\mathcal{A},\mathcal{V}}(\lambda)$ experiment. The adversary first calls $\mathcal{O}\mathsf{init}$ and then can call Oboard and $\mathsf{OreceiptLR}$ as much as it wants. Finally, the adversary calls $\mathcal{O}\mathsf{tally}$, receives the result of the election and must return its guess, which is the output of the experiment.

Intuitively, this security is reminiscent of BPRIV [8] with an additional $\mathcal{O}\mathsf{receiptLR}$ oracle. This oracle enables the adversary to cast ballots on behalf of the voter, as long as these two ballots share the same trace.

More precisely we consider two experiments parametrized by the value β. In each experiment, there are two bulletin boards, PB_0 and PB_1. The goal of the adversary is to distinguish if it interacts with the experiment using $\beta = 0$ or $\beta = 1$. To do so, it has the view of PB_β (through the $\mathcal{O}\mathsf{board}$) oracle and can cast two ballots ballot_0 to PB_0 and ballot_1 to PB_1. Those ballots need to have the same trace (otherwise it would be trivial to distinguish the boards) and are processed (through a rerandomization for example) before being added on the boards. Finally, the adversary has access to the result of the tally of PB_0 as well as the proof that the result is valid. In the case of $\beta = 1$, this proof is simulated as the result is inconsistent with the ballots on PB_1.