Traceable Receipt-Free Encryption

Henri Devillez, Olivier Pereira, and Thomas Peters

UCLouvain – ICTEAM – Crypto Group
B-1348 Louvain-la-Neuve – Belgium

Abstract. CCA-like game-based security definitions capture confidentiality by asking an adversary to distinguish between honestly computed encryptions of chosen *plaintexts*. In the context of voting systems, such guarantees have been shown to be sufficient to prove ballot privacy (Asiacrypt'12). In this paper, we observe that they fall short when one seeks to obtain receipt-freeness, that is, when corrupted voters who submit chosen *ciphertexts* encrypting their vote must be prevented from proving how they voted to a third party.
Since no known encryption security notion can lead to a receipt-free ballot submission process, we address this challenge by proposing a novel publicly verifiable encryption primitive coined Traceable Receipt-free Encryption (TREnc) and a new notion of traceable CCA security filling the definitional gap underlined above.
We propose two TREnc instances, one generic achieving stronger guarantees for the purpose of relating it to existing building blocks, and a dedicated one based on SXDH. Both support the encryption of group elements in the standard model, while previously proposed encryption schemes aiming at offering receipt-freeness only support a polynomial-size message space, or security in the generic group model.
Eventually, we demonstrate how a TREnc can be used to build receipt-free protocols, by following a standard blueprint.
Keywords. New primitive, public-key encryption, receipt-freeness.

Table of Contents

1 Introduction

A protocol offers receipt-freeness when players are unable to demonstrate to a third party which input they provided during a protocol execution. The need for receipt-freeness is most acute in order to prevent vote selling in the context of elections [7], which is our motivating application, .

Receipt-free voting. In voting protocols, the random coins used by the voters can often be used as a receipt. For instance, in the famous protocol by Cramer et al. [19], of which a variant is used by the IACR in its own elections, a voter encrypts his vote with the election public key, and the resulting ciphertext is posted on a public bulletin board in order to support the verifiability of the election. If the voter decides to reveal to a third party the randomness used in the encryption process, that party can re-encrypt the claimed vote intent with the randomness provided by the voter and verify that the resulting ciphertext appears on the bulletin board: the randomness used for encryption is, in effect, a receipt for the vote.

Since the seminal work of Benaloh [7], numerous protocols explored mechanisms that would guarantee that the random coins used by a voter are insufficient to explain his ballot as it is posted on the bulletin board for the needs of verification. In a first line of works [7,36,23], every possible voting choice is encrypted, the resulting ciphertexts are rerandomized and shuffled by the election authorities and made available to the voter. Furthermore, the permutations applied during the shuffle are also transmitted to the voter using secure channels. The voter then picks the ciphertext encoding his choice, and submits it for display on the bulletin board. Such a protocol guarantees that the voter ignores the randomness used to encrypt his ballot, and the protocol is designed in such a way that the voter is unable to prove which permutation he received, typically using designated-verifier zero-knowledge proofs. Such protocols are however quite demanding in terms of resources, as they require to encrypt a number of ciphertext proportional to the number of voting options, and a communication bandwidth to the voters that is proportional to the number of authorities. The more recent protocol of Kiayias et al. [26] faces similar challenges in terms of complexity, and also only considers a weaker form of receipt-freeness that focuses on voters preparing their ballot honestly.

More recently, Blazy et al. [10] proposed a simpler voting flow supporting receipt-freeness based on signatures on randomizable ciphertexts (SRC): the voters encrypt their vote and sign the resulting ciphertext, which is then transmitted to a re-encryption authority that re-randomizes the ciphertext, adapts the signature accordingly and posts the result on the bulletin board. The voter remains able to verify that a vote with a valid signature is posted on the board on his behalf, but is unable to explain the vote content thanks to the re-randomization step. Furthermore the SRC guarantees that the content of the encrypted ballot cannot be modified during the re-encryption process. This approach was further refined by Chaidos et al. [14], who also propose a simple game-based definition of receipt-freeness, which we adopt here, and more efficient SRCs keep being proposed [14,5].

This approach makes the ballot submission process asymptotically optimal for the voter, in the sense of Cramer et al. [19]: the protocol complexity for the voter becomes logarithmic in the number of voting options and independent of the number of election authorities, contrary to a dependency that is at least linear in both these factors when the approaches of [7,36,23,26] are used.

Receipt-free ballot submission. These works, by offering a simple ballot submission process in one pass, raise the natural question of identifying a public key encryption primitive that would support a receipt-free ballot submission process. Such a primitive would support a modular analysis of voting protocols that would be built around it, including various tallying approaches (based on mix-nets and homomorphic tallying for instance), and approaches to individual verifiability (based on the so-called Benaloh challenge [6] or on code voting for example [16]).

This question has been answered in the context of private (rather than receipt-free) ballot submission: it is well-known that a CCA-secure encryption scheme can be used to obtain a private ballot submission, a requirement that can be relaxed to NM-CPA security when the tally takes place in a single decryption round [20,39,9].

These works highlight the importance of some form of non-malleability in a submission process. From a practical point of view, non-malleability is needed in order to be able to detect (and prevent) non-independent ballot submissions (e.g., ballot copies) that would violate the privacy of the vote. From a technical point of view, security proofs require the availability of a decryption oracle used to extract the votes submitted by the adversary.

CCA security is however problematic in the context of receipt-free ballot submission, since we need to be able to re-randomize encrypted votes, so that the voter cannot explain the vote content anymore. The exploration of CCA-like security notions that would support some form of controlled malleability has been a fertile research area, which resulted in the definition of the notions of replayable-CCA (RCCA) security[13], homomorphic-CCA (HCCA) security [34], and controlled-malleable CCA (CM-CCA) security [15] for instance. As far as we know, all these works rely on the same CCA blueprint, in which an adversary submits one or more messages to a challenger, who answers either with a honest encryption of the messages or with something else, and the adversary must decide what he received with the help of a decryption oracle that accepts to decrypt any ciphertext that is not "recognizably" related to the challenge ciphertexts. The same holds in any other encryption primitives with CCA-like security with enhanced decryption capabilities. While they give more flexible ways to decrypt ciphertexts (based on identities, attributes and so on [37,24,35,11]), the challenge ciphertext is computed when the adversary sends a chosen *message*.

This blueprint is however inadequate when turning to encryption schemes that would support the design of protocols that support receipt-freeness: in such a setting, we need to consider an adversary who sends chosen *ciphertexts*, that may not be computed as a random encryption of a plaintext vote.

Our contributions

1) TCCA security. In this work, we investigate for the first time the implication of defining the notion of *traceable* CCA security (TCCA), a CCA-like security notion in which adversarially-chosen ciphertexts are submitted in the challenge phase. The challenge ciphertext is produced by *randomizing* one ciphertext or another, and we recognize derivatives of the challenge ciphertext thanks to a non-malleable public *trace* which is present in any ciphertext. To avoid trivial attacks, both ciphertexts given in the challenge phase must trace to each other, i.e., they must have the same trace.

This makes it possible for voters to submit a ciphertext of their choice, which will then be re-randomized by an authority, and can still be tracked by the voters using the trace.

For honestly produced ciphertexts, our security notion also implies traditional confidentiality properties, so that ballot privacy remains guaranteed should the re-randomizing authority be corrupted. So, non-malleability really serves two purposes here: *(1)* it guarantees that the re-randomizing authority cannot produce a ciphertext that would be related to a honestly produced one and have a different trace (which would violate ballot privacy), and *(2)* it guarantees that the re-randomizing authority cannot produce a ciphertext that would have the same trace as a given one but would decrypt to a different plaintext.

2) TREnc. We introduce Traceable Receipt-free Encryption (TREnc) as a new primitive with the following features:

- *Traceability.* Honestly generated ciphertexts are traceable in the sense that it is infeasible to modify the encrypted message;
- *Randomizability.* Valid ciphertexts are fully re-randomizable, up to the trace;
- *TCCA security.* Given a pair of ciphertexts that trace to each other, it is unfeasible to guess which one is randomized, even with access to a decryption oracle which decrypts any ciphertexts that do not trace to the challenge ciphertext, except before the challenge phase.

We also provide:

1. A generic TREnc that can be instantiated from existing building blocks that offer security in the standard model, and whose CRS is public-coin;
2. A pairing-based TREnc under the SXDH assumption in the standard model, where the public key only contains 13 first-source group elements and 6 second-source group elements, and the ciphertext contains 13 first-source group elements and 5 second-source group elements.

Both approaches improve on the state of the art: the previous SRC-based solutions either require costly bit-by-bit encryption [10,14], or only offer security in the generic group model [5].

3) A TREnc based voting scheme. Eventually, we show how to turn a TREnc into a simple voting scheme in a generic way, following the *Enc2Vote* blueprint previously used to turn a CCA-secure encryption scheme into a private voting scheme [9].

We demonstrate that the resulting voting scheme satisfies a notion of receipt-freeness that is equivalent in spirit to the one of Chaidos et al. [14], but fixes a small technical issue in that definition that makes their security game trivial to win (making it impossible to build a protocol that is receipt-free according to their definition).

Other related works. We focus on offering receipt-freeness in the context of voting, which is the context in which receipt-freeness was introduced [7], and which remains the main application context in which receipt-freeness is desired. Voting can however be seen as a special type of secure function evaluation protocol, in which specific tallying functions are evaluated and, as such, the notion of receipt-freeness, and the related notion of coercion-resistance have also been defined in the general multi-party computation setting [12,33,38,4]. We keep our focus on the voting context in order to clarify various design choices that are most meaningful in the voting setting compared to the general MPC setting: our primitive is targetted for a ballot submission process in which voters submit their ballot in one pass and do not communicate with each other, contrary to most MPC protocols, and we design mechanisms in which the ballot submission process can be fast, even on devices with limited computational power, while the verification of an election may require a longer period of time and use a dedicated computing infrastructure. Despite our focus on voting, it may be the case that TREnc mechanisms find applications in other contexts.

2 Traceable Receipt-Free Encryption

We propose a new public key encryption primitive and associated security notions that would support the receipt-free submission of votes in a protocol. As a first task, we identify the fundamental ingredients that are needed for our new encryption primitive.

An encryption scheme. We expect voters to submit their vote in an encrypted form, in order to guarantee the privacy of the votes.

Receipt-free encryption. Voters willing to sell their vote may choose to submit an arbitrary encrypted vote, which may be in the range of honestly produced ciphertexts but sampled according to a different distribution, or even just a sequence of bits that would not be within the range of the encryption mechanism. By deviating from the normal encryption process, the voter hopes to obtain a receipt that could be used to demonstrate his vote intent to a third party.

If the encrypted vote that is tallied is produced by the voter only, then the voter will always have a receipt: the random coins used to encrypt the ballot. In order to avoid this, we rely on the existence of a semi-trusted authority: that authority will be trusted to prevent a dishonest voter from obtaining a receipt for his vote, but will not be trusted for the correctness of the election result, and will not be trusted for the privacy of votes encrypted by honest voters.

Concretely, in order to achieve receipt-freeness, this semi-trusted authority tests the validity of a voter submitted ciphertext (without the need of any secret key) and re-randomizes every valid ciphertext before posting it on a public bulletin board.

Traceable Receipt-Free Encryption. In order to make it possible for a voter to check that his ballot has not been unduly modified by this semi-trusted re-randomizing authority, it must be possible to extract a *trace* from any valid ciphertext. A honest re-randomization process would keep the trace is unchanged, hence making ciphertexts *traceable*, while no corrupted authority should be able to modify a ciphertext in such a way that it would decrypt to a different vote while keeping the trace unchanged.

Furthermore, we need to make sure that this trace cannot serve as a receipt for the vote. In order to make sure that it is the case, we split the encryption process in two steps, that guarantee that any trace can be associated to any possible vote intent. Concretely, an encryption starts with the generation of a secret *link key*, which is then used, together with the encryption public key, to encrypt any possible vote. This guarantees that, even if a voter leaks the link key associated to his ballot as a receipt, the ballot could still encrypt any vote.[1]

2.1 Syntax

We now have the ingredients that we need to define a Traceable Receipt-Free Encryption scheme, or TREnc.

Definition 2.1 (Traceable Receipt-Free Encryption). *A* Traceable Receipt-Free Encryption *scheme (TREnc) is a public key encryption scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *that is augmented with a 5-tuple of algorithms* $(\mathsf{LGen}, \mathsf{LEnc}, \mathsf{Trace}, \mathsf{Rand}, \mathsf{Ver})$*:*

- $\mathsf{LGen}(\mathsf{pk}; r)$*: 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.*

In many cases, we will omit the randomness r *from our notations. It is then assumed that it is selected uniformly at random.*

We require several correctness properties from the additional algorithms of a TREnc. The first requires that encrypting a message m by picking a link key lk using LGen and computing $\mathsf{LEnc}(\mathsf{pk}, \mathsf{lk}, m)$ produces a ciphertext that is identically distributed to a fresh encryption of m using Enc. The second requires that the Trace of a ciphertext does not depend on the message that is encrypted. The third requires that randomizing a ciphertext does not change the corresponding plaintext neither the corresponding trace. The last requires that every honestly computed ciphertext passes the verification algorithm.

Definition 2.2 (TREnc correctness). *We require that a TREnc scheme satisfies the following correctness requirements.*

Encryption compatibility *For every* pk *in the range of* Gen *and message* m*, the distributions of* $\mathsf{Enc}(\mathsf{pk}, m)$ *and* $\mathsf{LEnc}(\mathsf{pk}, \mathsf{LGen}(\mathsf{pk}), m)$ *are identical.*

[1] Of course, this also means that, if a corrupted re-randomizing authority obtains a voter's secret link key (e.g., by corrupting the voter's voting client), then it might be able to produce a ciphertext that encrypts a different vote intent but would still trace to the original voter trace. Just as other attacks related to corrupted voting clients, such attacks can be prevented by traditional continuous ballot testing procedures [6], in which a voter would have the option to ask an authority to spoil a ballot posted on the bulletin board, which would then be verifiability decrypted for verification, and later replaced by a fresh new ballot produced by the voter, using a fresh link key.

Link traceability *For every* pk *in the range of* Gen*, every* lk *in the range of* $\mathsf{LGen}(\mathsf{pk})$*, the encryptions of every pair of messages* (m_0, m_1) *trace to the same trace, that is, it always holds that* $\mathsf{Trace}(\mathsf{pk}, \mathsf{LEnc}(\mathsf{pk}, \mathsf{lk}, m_0)) = \mathsf{Trace}(\mathsf{pk}, \mathsf{LEnc}(\mathsf{pk}, \mathsf{lk}, m_1))$*.*

Publicly Traceable Randomization *For every* pk *in the range of* Gen*, every message* m *and every* c *in the range of* $\mathsf{Enc}(\mathsf{pk}, m)$*, we have that* $\mathsf{Dec}(\mathsf{sk}, c) = \mathsf{Dec}(\mathsf{sk}, \mathsf{Rand}(\mathsf{pk}, c))$ *and* $\mathsf{Trace}(\mathsf{pk}, c) = \mathsf{Trace}(\mathsf{pk}, \mathsf{Rand}(\mathsf{pk}, c))$*.*

Honest verifiability *For every* pk *in the range of* Gen *and every messages* m*, it holds that* $\mathsf{Ver}(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, m)) = 1$

2.2 Security definitions

Verifiability We require several security properties from a TREnc. Our first property is fairly standard: a TREnc is verifiable if the Ver algorithm guarantees that a ciphertext is within the range of Enc. In other words, the ciphertext can be explained by some message m, some link key lk, and some coins, even if they are not easily computable.

Definition 2.3 (Verifiability). *A TREnc is* verifiable *if for every* PPT *adversary, the following probability is negligible in* λ*:*

$$\Pr[\mathsf{Ver}(\mathsf{pk}, c) = 1 \textit{ and } c \notin \mathsf{Enc}(\mathsf{pk}, \cdot) | (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda); c \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{sk})].$$

TCCA *security.* We now turn to our central security definition, *security against traceable chosen ciphertexts attacks*, or TCCA security, which differs from all existing CCA-like notions by letting the adversary submit pairs of ciphertexts instead of pairs of messages, reflecting that we need security in front of adversarially chosen ciphertexts. In the TCCA security game (Fig. 1), the adversary receives the public key and has access to a decryption oracle, as usual. It then submits a pair of ciphertexts that must be valid and have identical traces. One of the ciphertexts is randomized and returned to the adversary, who must decide which one it is. After receiving this challenge ciphertext, the adversary can still query the decryption oracle, but only on ciphertexts that have a trace different of his challenge ciphertext. So, the challenger must faithfully decrypt pre-challenge ciphertexts that have the same trace as the challenge ciphertext. Looking ahead, this decryption capability offers an easy but necessary mean allowing simulating the result of an election when proving receipt-freeness.

TCCA security guarantees that, if a voter submits a ciphertext that is randomized before it is posted on a public bulletin board, then the resulting ciphertext becomes indistinguishable from any other ciphertext that would have the same trace, and 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.

Definition 2.4 (TCCA)**.** *A TREnc is* TCCA *secure if for every* PPT *adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *the experiment* $\mathsf{Exp}^{\mathrm{tcca}}_{\mathcal{A}}(\lambda)$ *defined in Figure 1 (left) returns* 1 *with a probability negligibly close in* λ *to* $\frac{1}{2}$*.*

It is naturally possible to write a multi-challenge version of the $\mathsf{Exp}^{\mathrm{tcca}}_{\mathcal{A}}(\lambda)$ experiment, which we call q-TCCA, in which the adversary can submit q pairs of ciphertexts. This leads to an equivalent

```
$\mathsf{Exp}^{\mathrm{tcca}}_{\mathcal{A}}(\lambda)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{Gen}(1^\lambda)$
$(c_0, c_1, \mathsf{st}) \leftarrow\$ \mathcal{A}_1^{\mathsf{Dec}(\cdot)}(\mathsf{pk})$
$b \leftarrow\$ \{0,1\}$
if $\mathsf{Trace}(\mathsf{pk}, c_0) \neq \mathsf{Trace}(\mathsf{pk}, c_1)$ or
$\mathsf{Ver}(\mathsf{pk}, c_0) = 0$ or $\mathsf{Ver}(\mathsf{pk}, c_1) = 0$ then return $b$
$c^\star \leftarrow\$ \mathsf{Rand}(\mathsf{pk}, c_b)$
$b' \leftarrow\$ \mathcal{A}_2^{\mathsf{Dec}^\star(\cdot)}(c^\star, \mathsf{st})$
return $b' = b$
```

```
$\mathsf{Exp}^{\mathrm{trace}}_{\mathcal{A}}(\lambda)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{Gen}(1^\lambda)$
$(m, \mathsf{st}) \leftarrow\$ \mathcal{A}_1(\mathsf{pk}, \mathsf{sk})$
$c \leftarrow\$ \mathsf{Enc}(\mathsf{pk}, m)$
$c^\star \leftarrow\$ \mathcal{A}_2(c, \mathsf{st})$
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. TCCA and trace experiments. In the TCCA experiment, $\mathcal{A}_2$ has access to a decryption oracle $\mathsf{Dec}^\star(\cdot)$ which, on input c, returns $\mathsf{Dec}(c)$ if $\mathsf{Trace}(\mathsf{pk}, c) \neq \mathsf{Trace}(\mathsf{pk}, c^\star)$ and test otherwise.

definition, as demonstrated in Appendix B. We also stress that in the challenge query the adversary may know the random coins underlying c_0 and c_1 and may have drawn them from a specific secret distribution. The randomization leading to the challenge ciphertext $c^\star$ should thus erase any subliminal information binding $c^\star$ to the message in c_b. This definition introduces some technical difficulty when it comes to proving the TCCA security as it becomes harder to program the public key to ease the transition toward a game where we are able to inject an independent message in the plaintext in an undetectable way. Indeed, we have no clue at the setup time about the distribution of (c_0, c_1) and their common trace while the emulation of $\mathsf{Rand}(\mathsf{pk}, c_b)$ must preserve it without even knowing the underlying link keys.

TCCA security is reminiscent of the notion of publicly detectable replayable-CCA (pd-RCCA) security proposed by Canetti et al. [13]. The pd-RCCA security game is essentially the same as the CCA game, except for two main differences: a publicly computable equivalence relation is defined on ciphertexts and, after the challenge ciphertext has been received, the challenger will refuse to decrypt any ciphertext that is equivalent to this challenge ciphertext. Furthermore, ciphertexts that are in the same equivalence class must decrypt to the same message (for completeness, the full definition is available in Appendix A.2). The pd-RCCA security game looks appealing in the context of voting, because it captures this idea of having the possibility to re-randomize ciphertexts while also keeping a trace that could be detected through the equivalence relation. And, indeed, RCCA-secure encryption has been used in previous proposals of receipt-free voting schemes [14].

There are three central differences, though, which motivate the introduction of the TCCA security game.

- The challenge ciphertexts of the pd-RCCA security game are always honestly computed and, as such, pd-RCCA security does not offer any guarantee in front of maliciously produced ciphertexts, as it would be the case when a voter tries to obtain a receipt for his vote.
- Contrary to pd-RCCA security, it can be observed that TCCA security says nothing about the hiding property of the Enc algorithm, since the adversary must distinguish based on outputs of Rand. An extreme case could define Enc as the identity function, Trace as mapping to a single constant trace, and Rand actually performing the encryption work, and this could still offer a TCCA secure scheme. The confidentiality requirements on Enc will be handled through the traceability and strong randomization properties below.

– There is no requirement for TCCA security that trace equivalent ciphertexts decrypt to the same message: a single link key can be used to encrypt any message, and all the resulting ciphertexts would have the same trace (by the link traceability correctness property). We recall that this non-binding feature is essential for receipt-free voting.

As such, TCCA security is not comparable to pd-RCCA security. We show in Appendix B that, under (different) additional conditions, implications can be proven in both directions for a natural variant of pd-RCCA security adapted to TREnc schemes.

Traceability and Strong Randomization. While TCCA security relates to a model in which the voting client may be corrupted but the re-randomization server is honest, we now focus on two central properties that are important when the voting client is honest and the re-randomization server might be corrupted.

The traceability property guarantees to the sender of a honestly encrypted message that no efficient adversary would be able to produce another ciphertext that traces to the same trace and would decrypt to a different message, even if the adversary knows the secret decryption key. So, even if a TREnc offers some form of ciphertext malleability, its traceability implies the non-malleability of the plaintexts. This is an important feature for the verifiability of a voting system: as long as the link key used to encrypt a vote remains secret, and the voter submits a single ciphertext encrypted with that link key, the voter is guaranteed that any ciphertext that would trace to his original ciphertext encrypts his original vote. (But, of course, using the link key, it remains possible to produce ciphertexts with the same trace that would decrypt to any vote.)

Definition 2.5 (Traceability). *A TREnc is* traceable *if for every* PPT *adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, the experiment* $\mathsf{Exp}^{\mathrm{trace}}_{\mathcal{A}}(\lambda)$ *defined in Figure 1 (right) returns* 1 *with a probability negligible in* λ*.*

The second property, strong randomization, requires that the output of the Rand algorithm applied to any valid ciphertext is distributed just as a random encryption of the same message with the same link key.

Definition 2.6 (Strong Randomization). *A TREnc is* strongly randomizable *if for every* $c \in \mathsf{LEnc}(\mathsf{pk}, \mathsf{lk}, m)$ *with* pk *in the range of* Gen *and* lk *in the range of* $\mathsf{LGen}(\mathsf{pk})$*, the following computational indistinguishability relation holds:*

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

Requiring strong randomization together with TCCA security guarantees that Enc actually hides messages. CPA security comes easily: when the CPA adversary sends (m_0, m_1) to the TCCA adversary, the TCCA adversary can encrypt the 2 messages using a single random link key and send them to the TCCA challenger, which will return a randomization of one of them. Strong randomization guarantees that this is distributed exactly like an encryption of one of the two messages, and we can send the result to the CPA adversary, who will then offer the answer expected for the TCCA game. We show a stronger implication to RCCA security in Appendix B.2.

3 Towards a generic TREnc

We are now interested in exploring how a TREnc could be designed from existing tools. The core TREnc security feature comes from the TCCA security game, in which the adversary submits a pair of ciphertexts with identical traces and receives a re-randomization of one of them. If we want relate this game to a more standard RCCA-style security definition in which the adversary submits a pair of plaintext and receives an encryption of one of them, we need to be able to translate a re-randomization query on two ciphertexts into an encryption query on the two corresponding plaintexts. But there is an additional constraint that needs to be satisfied: the ciphertext resulting from the encryption query needs to have the same trace as the original ciphertexts. In other words, we need to be able to decrypt the challenge ciphertexts from the TCCA game, but also to extract the link key that they contain. We capture this last idea in an augmented version of a TREnc, which we call *extractable TREnc*.

3.1 Extractable TREncs

Essentially, an extractable TREnc makes it possible to produce encryption keys together with a trapdoor using a $\mathsf{TrapGen}$ algorithm. Using that trapdoor, it becomes possible to extract, from any ciphertext, a link key that makes it possible to produce new ciphertexts with the same trace as the original one. This in turn implies the possibility to break the traceability of the scheme.

Definition 3.1 (Extractable TREnc). *An extractable TREnc is a TREnc with two additional algorithms* $\mathsf{TrapGen}$ *and* LExtr*:*

- $\mathsf{TrapGen}(1^\lambda)$*: The trapdoor generation algorithm takes as input the security parameter and outputs a tuple of public/secret/trapdoor keys* $(\mathsf{pk}, \mathsf{sk}, \mathsf{tk})$*. We require the distribution of the* $(\mathsf{pk}, \mathsf{sk})$ *pairs produced by* $\mathsf{TrapGen}(1^\lambda)$ *to be identical to the one of the outputs of* $\mathsf{Gen}(1^\lambda)$*.*
- $\mathsf{LExtr}(\mathsf{tk}, c)$*: The link extraction algorithm takes as input the trapdoor key and a ciphertext and returns a link key* lk *such that, if* c *is in the range of* $\mathsf{Enc}(\mathsf{pk}, \cdot)$ *with* pk *in the range of* Gen*, then* c *is in the range of* $\mathsf{LEnc}(\mathsf{pk}, \mathsf{lk}, \cdot)$*.*

It is fairly natural to require that ciphertexts can only be consistent with one single link key, hence guaranteeing a unique link key extraction.

Definition 3.2 (Unique Extraction). *An extractable TREnc has* unique extraction *if, for every* $(\mathsf{pk}, \mathsf{sk}, \mathsf{tk})$ *in the range of* $\mathsf{TrapGen}$ *and* lk *in the range of* $\mathsf{LGen}(\mathsf{pk})$*, we have that:*

- $\mathsf{LExtr}(c, \mathsf{tk}) = \mathsf{lk}$ *whenever* $c \in \mathsf{LEnc}(\mathsf{pk}, \mathsf{lk}, \cdot)$*;*
- $\mathsf{LExtr}(c_0, \mathsf{tk}) = \mathsf{LExtr}(c_1, \mathsf{tk})$ *whenever we have* $\mathsf{Trace}(\mathsf{pk}, c_0) = \mathsf{Trace}(\mathsf{pk}, c_1)$ *and* $c_0, c_1 \in \mathsf{Enc}(\mathsf{pk}, \cdot)$*.*

3.2 A TREnc flavored variant of pd-RCCA security

Based on an extractable TREnc, we now propose an RCCA-like security definition, pd*-RCCA-security, which shares much of the spirit of the pd-RCCA notion of Canetti et al. [13], but is rather tailored as a useful intermediary notion for achieving TCCA security: we will show that any pd*-RCCA-secure extractable TREnc is also TCCA secure. Eventually, we will show how to achieve pd*-RCCA-security from existing tools.

Definition 3.3 (pd*-RCCA). *An extractable TREnc is* pd*-RCCA-*secure if for any* PPT *adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the experiment $\mathsf{Exp}_{\mathcal{A}}^{\text{pd}^\star\text{-rcca}}(\lambda)$ in Figure 2 returns* 1 *with a probability negligibly close on λ to $\frac{1}{2}$.*

```
$\underline{\mathsf{Exp}_{\mathcal{A}}^{\text{pd}^\star\text{-rcca}}(\lambda)}$
$(\mathsf{pk}, \mathsf{sk}, \mathsf{tk}) \leftarrow\$ \mathsf{TrapGen}(1^\lambda)$
$(m_0, m_1, \mathsf{lk}, \mathsf{st}) \leftarrow\$ \mathcal{A}_1^{\mathsf{Dec}(\mathsf{sk},\cdot),\mathsf{LExtr}(\mathsf{tk},\cdot)}(\mathsf{pk})$
$b \leftarrow\$ \{0,1\}$
**if** $\mathsf{lk} = \perp$ **then** $c^\star \leftarrow\$ \mathsf{Enc}(\mathsf{pk}, m_b)$
**else** $c^\star \leftarrow\$ \mathsf{LEnc}(\mathsf{pk}, \mathsf{lk}, m_b)$
$b' \leftarrow\$ \mathcal{A}_2^{\mathsf{Dec}^\star(\mathsf{sk},\cdot),\mathsf{LExtr}(\mathsf{tk},\cdot)}(c^\star, \mathsf{st})$
**return** $b = b'$
```

Fig. 2. pd*-RCCA experiment. Here, $\mathsf{Dec}^\star(\mathsf{sk}, c)$ is a decryption oracle that returns test if $\mathsf{Trace}(\mathsf{pk}, c) = \mathsf{Trace}(\mathsf{pk}, c^\star)$ and $\mathsf{Dec}(\mathsf{sk}, c)$ otherwise.

Just as in the pd-RCCA security definition, our adversary receives a public key, then can make decryption queries, make a challenge query on a pair of plaintexts, receive an encryption $c^\star$ of one of them, and then make more decryption queries, provided that they are not about ciphertexts that are equivalent to $c^\star$. Here the notion of equivalent ciphertext is defined by ciphertexts with identical traces, which does not imply that they decrypt to the same plaintext, contrary to the compatibility requirement of pd-RCCA security. The extra features of pd*-RCCA security, which come naturally in the context of an extractable TREnc, are that:

- On top of having access to a decryption oracle, the adversary has access to a LExtr oracle that gives him the possibility to extract the link key from any ciphertext.
- During his challenge query, the adversary can provide a link key on top of its two plaintexts: the challenge ciphertext will then be computed using that link key.

As announced, a pd*-RCCA-secure and strongly randomizable extractable TREnc is also TCCA-secure.

Theorem 3.4. *If a TREnc scheme T is extractable, strongly randomizable, and* pd*-RCCA-*secure, then it is* TCCA *secure. More precisely, if the advantages of any PPT adversary at strong randomization and* pd*-RCCA *experiment are respectively bounded by ε_{SR} and ε, then for any* PPT *adversary $\mathcal{A}$, we have $Pr[\mathsf{Exp}_{\mathcal{A},T}^{\text{tcca}}(\lambda) = 1] \leq \frac{1}{2} + \varepsilon_{SR} + \varepsilon$.*

Proof. (See Appendix B.1 for details.) The decryption queries from the TCCA adversary are forwarded to the pd*-RCCA challenger. When the TCCA adversary makes his challenge query on (c_0, c_1), the reduction obtains the corresponding link key and plaintexts by querying the pd*-RCCA challenger, and sends them as pd*-RCCA challenge. The resulting ciphertext is correctly distributed thanks to strong randomizability, and has the correct trace thanks to the extractability. The winning probability of the TCCA adversary is then negligibly close to the winning probability of the resulting pd*-RCCA adversary. □

3.3 Building a pd*-RCCA-secure extractable TREnc

We are now ready to build a TREnc. As a first natural building block, we use a signature on randomizable ciphertexts (SRC), as introduced by Blazy et al. [10]. In an SRC, any signed ciphertext can be publicly re-randomized, and the signature can be publicly adapted so that it remains valid for the new ciphertext.

We can easily obtain the structure of a TREnc from an SRC by defining the LGen function as setting lk as a fresh signing key for the SRC, and the LEnc function as encrypting the plaintext using a randomizable encryption scheme, then signing that ciphertext using lk. The trace of a ciphertext would then be the signature verification key.

This offers a promising skeleton, but it is not sufficient to obtain pd*-RCCA security: as it is, the adversary could simply remove the signature from the challenge ciphertext, sign that ciphertext with a fresh key in order to obtain a different trace, and ask for the decryption of the result, which would be granted.

A natural solution to this problem is to link the trace to the ciphertext using tag-based encryption [27] mechanism. In a tag-based encryption scheme, the encryption and decryption functions take an arbitrary tag as an extra input, and the decryption of a ciphertext with an incorrect tag will fail. We rely on the standard notion of weak-CCA security for tag based encryption [32], which is the CCA security game excepted that the challenge ciphertext is produced using an *adversarially chosen* tag, and that no decryption query can be made using that tag (only) *after* the challenge phase. This security game nicely fits our pd*-RCCA security game, in which the trace derived from the link key submitted by the adversary can be used as a tag, and guarantees that no ciphertext can be modified in such a way that it successfully decrypts with a tag that is different of the original one. We note that we must be able to decrypt pre-challenge queries that already contains the "challenge tag" of the adversary, which prevents us from only relying on (weak) selective-tag security.

But we still need to be able to extract the link key from a tag-based ciphertext. This can be done fairly easily, by augmenting our encryption process with the requirement to encrypt the link key using a randomizable CPA-secure encryption scheme, and to add a randomizable ZK proof that the encrypted link key is indeed the one that is used as tag for the tag-based encryption. Extraction would then simply proceed by decrypting that CPA ciphertext. (In particular, it does not rely on any extraction property of the ZK proof system: we just need its soundness.)

So, to summarize we build an extractable TREnc from the following ingredients:

- A randomizable weakly CCA secure tag-based encryption scheme (TBGen, TBEnc, TBDec).
- An SRC compatible with the tag-based encryption scheme, which includes a signature scheme $(\mathsf{SGen}, \mathsf{Sign}, \mathsf{SVer})$.

– A randomizable CPA secure public key encryption scheme $(\mathsf{EGen}, \mathsf{EEnc}, \mathsf{EDec})$.
– A randomizable NIZK proof system $(\mathsf{Prove}, \mathsf{VerifyProof})$ that, on input (c_{tbe}, c_{extr}) and associated public keys, demonstrates that c_{extr} is an encryption with EEnc of the signing key whose corresponding verification key has been used as a tag in order to compute c_{tbe} using TBEnc.

And the blueprint of our TREnc is as follows:

– $\mathsf{TrapGen}$ uses TBGen to produces a key pair $(\mathsf{tpk}, \mathsf{tsk})$, and EGen to produce a key pair $(\mathsf{e}_{extr}, \mathsf{d}_{extr})$. It returns $\mathsf{pk} = (\mathsf{tpk}, \mathsf{e}_{extr})$, $\mathsf{sk} = \mathsf{tsk}$ and $\mathsf{tk} = \mathsf{d}_{extr}$.
– LGen sets lk as a signing key obtained from Sign. We assume that the corresponding signature verification key vk can be derived from lk.
– LEnc encrypts the message m as follows:
  - $c_{tbe} = \mathsf{TBEnc}(\mathsf{tpk}, \mathsf{lk}, m)$;
  - $\sigma = \mathsf{Sign}(\mathsf{lk}, c_{tbe})$;
  - $c_{extr} = \mathsf{EEnc}(\mathsf{e}_{\mathsf{extr}}, \mathsf{lk})$;
  - $\pi = \mathsf{Prove}(c_{tbe}, c_{extr}; \mathsf{lk})$

  The ciphertext is made of these 4 elements, together with vk.
– Dec returns $\mathsf{TBDec}(\mathsf{tsk}, \mathsf{vk}, c_{tbe})$
– Trace returns vk.
– Rand re-randomizes c_{tbe}, adapts σ accordingly, re-randomizes c_{extr}, and re-randomizes and adapts π.
– Ver accepts a TREnc ciphertext if the two ciphertexts that it contains are valid, if the signature is valid, and if the ZK proof verifies.
– LExtr returns $\mathsf{EDec}(\mathsf{tsk}, c_{extr})$.

A complete description of this generic TREnc, together with proofs of its security, is available in Appendix C. This section showed that the notion of extractable TREnc offers a convenient companion for a TREnc: it is possible to build an extractable TREnc from relatively common, yet strong, building blocks, and the proof of TCCA security of this TREnc comes relatively easily because we can design a pd-RCCA-like security notion for extractable TREnc that implies our new TCCA security notion. The resulting construction is however expected to be fairly expensive since, in the standard model, all known instantiations of the building blocks (see Section C, for instance) relies on a bit-by-bit decomposition of the message or the secret singing key of which the ciphertext must contain a (malleable) ZK proof of. Nevertheless, providing this extractability feature is an artifice for the construction that is not necessary for the security of the TREnc, but as far as we know there is no obvious generic construction leading to a TREnc without extractability. In the next section we turn to the construction of an ad-hoc efficient instance of a TREnc based on a standard computational assumption that also avoid the costly bit-by-bit decomposition.

4 Pairing-Based Construction under SXDH

This section provides a secure TREnc in the standard model, only relying on the SXDH assumption and on a CRS. Contrary to our previous construction, this one is not extractable – extractability was just a convenience but does not offer and security benefit. This allows us to get a more efficient solution, here, in asymmetric bilinear groups. Moreover, our construction enjoys a short public-key and short ciphertexts as they only contain a constant number of group elements to encrypt a full group element, contrary to previous proposals that required to process the message bit by bit [10,14].

We first introduce the cryptographic assumptions on which we will rely, as well as the main existing building block that we will use: linearly homomorphic structure-preserving signatures.

4.1 Computational setting

We rely on an efficient Setup algorithm that generates common public parameters pp. Given a security parameter λ, $\mathsf{Setup}(1^\lambda)$ generates a bilinear group $\mathsf{pp} = (\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, p, e, g, \hat{g}, \hat{h})$ of prime order $p > 2^{\mathrm{poly}(\lambda)}$ for some polynomial poly, where $g \leftarrow\!\$\, \mathbb{G}$ and $\hat{g}, \hat{h} \leftarrow\!\$\, \hat{\mathbb{G}}$ are random 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 must be hard in both $\mathbb{G}$ and $\hat{\mathbb{G}}$. Following the Groth-Sahai standard notation, we also define the linear map $\iota : \mathbb{G} \to \mathbb{G}^2$ with $\iota : Z \mapsto (1, Z)$.

4.2 Linearly Homomorphic Structure-Preserving Signatures

A central tool for our efficient TREnc construction is linearly homomorphic structure-preserving signatures. The structure preserving [2][1] property makes it possible to sign messages that are group elements (and not just bits as in schemes based on the Waters signature), while the additional linearly homomorphic feature, introduced by Libert et al. [30], will be used to make the signatures randomizable while guaranteeing the non-malleability of the plaintext.

$\mathsf{Keygen}(\mathsf{pp}, n)$: 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 $\mathsf{pk} = \{\hat{g}_i\}_{i=1}^n \in \mathbb{G}^n$.

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

$\mathsf{SignDerive}(\mathsf{pk}, \{(\omega_i, \sigma^{(i)})\}_{i=1}^\ell)$: given pk as well as ℓ tuples $(\omega_i, \sigma^{(i)})$, parse $\sigma^{(i)}$ as $\sigma^{(i)} = \left(Z_i, R_i\right)$ for $i = 1$ to ℓ. Return the triple $\sigma = (Z, R) \in \mathbb{G}$, where $Z = \prod_{i=1}^\ell Z_i^{\omega_i}$, $R = \prod_{i=1}^\ell R_i^{\omega_i}$.

$\mathsf{Verify}(\mathsf{pk}, \sigma, (M_1, \ldots, M_n))$: 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

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

4.3 Intuition of our construction

To encrypt a message $m \in \mathbb{G}$, we combine a CPA encryption $\boldsymbol{c} = (c_0, c_1, c_2)$ of the form $c_0 = m \cdot f^\theta$, $c_1 = g^\theta$, $c_2 = h^\theta$ and and a randomizable publicly verifiable proof that $\log_g c_1 = \log_h c_2$, à la Cramer-Shoup. For that purpose, we can rely on the idea to include a one-time LHSP signature on top of $\boldsymbol{c}$ as first suggested in [30]. That means that the public key contains an LHSP signature Σ on (g, h) so that we can derive a signature on $(g, h)^\theta$ if indeed (c_1, c_2) lies in span$\langle(g, h)\rangle$ by computing $\pi = \Sigma^\theta$. Such a proof is quasi-adaptive [25] as the CRS depends on the language of which we have to prove membership. Here, the public key includes a CRS that contains a signature on the basis of the linear subspace span$\langle(g, h)\rangle$ of $\mathbb{G}^2$. Given $\boldsymbol{c}$ and the LHSP signature π one can easily randomize the ciphertext as follows: compute $\boldsymbol{c}' = \boldsymbol{c} \cdot (f, g, h)^{\theta'}$, and adapt the proof $\pi' = \pi \cdot \Sigma^{\theta'}$. While this solution is perfectly randomizable and the signing key allows to perfectly simulate the proof, it only provides a CCA1 security. Still, this technique has been enhanced to provide tag-based simulation-sound proof system which is reminiscent to building CCA-like secure encryption. The underlying technique is to generate a one-time key pair $(\mathsf{opk}, \mathsf{osk})$ of some one-time signature scheme that will be discussed in the next paragraph, and to define the tag as $\tau = H(\mathsf{opk})$, for some collision-resistant hash function H,[2] before computing π that (c_1, c_2) lies in span$\langle(g, h)\rangle$ based on τ. The ciphertext is then completed by signing $(\boldsymbol{c}, \pi)$ with osk, resulting in the ciphertext $(\boldsymbol{c}, \pi, \sigma, \mathsf{opk})$. A natural solution would be to borrow the first solution due to [31] but it only provides selective-tag simulation soundness. Since we will be using opk as the trace of our TREnc construction, the TCCA security implies that our underlying tag-based encryption must achieve tag-based weak CCA security, and selective-tag security is not enough (see Section C.2). Indeed, the tag $\tau^* = H(\mathsf{opk}^*)$ involved in the challenge ciphertext may be chosen by the adversary at any time. Furthermore, we must be able to answer *any* pre-challenge decryption queries, so even those that already used τ^*. That means that we cannot program the public key to embed τ^* that will help us to incorporate an SXDH instance in the computation of the challenge ciphertext. Fortunately, by including a signature Σ_u on $(g, h, 1, 1)$ and another signature Σ_v on $(1, 1, g, h)$ in the public key, given a tag τ, the computation of $\pi = (\Sigma_u^\tau \Sigma_v)^\theta$ due to [28] is an LHSP signature on $(c_1^\tau, c_2^\tau, c_1, c_2)$ which gives us the expected security and still enjoys a perfect randomizability, but for the given tag τ (and trace opk), only, which is still what we were looking for.

Now, we come back on the signature σ of $(\boldsymbol{c}, \pi)$. Usually, $(\mathsf{opk}, \mathsf{osk})$ is a key pair of a strongly unforgeable signature scheme providing non-malleability of ciphertext. However, we want to keep the malleability of the ciphertext as we want to be able to fully randomize it up to opk that will serve as our trace, but we also want to retain the non-malleability of the encrypted message m to satisfy traceability. Here again, in the standard model under SXDH, LHSP signature scheme comes in handy. If our fresh key pair $(\mathsf{opk}, \mathsf{osk})$ is generated from a one-time LHSP signature scheme, we can fix the message and preserve the randomizability of $(\boldsymbol{c}, \pi)$ by computing one-time LHSP signatures σ_1 on (g, c_0, c_1, c_2) and σ_2 on $(1, f, g, h)$. Like this, when we randomize $(\boldsymbol{c}, \pi, \sigma_1, \sigma_2)$ as $\boldsymbol{c}' = \boldsymbol{c} \cdot (f, g, h)^{\theta'}$, $\pi' = \pi \cdot (\Sigma_u^\tau \Sigma_v)^{\theta'}$ we can also adapt the signature $\sigma_1' = \sigma_1 \cdot \sigma_2^{\theta'}$ on $(g, c_0', c_1', c_2') = (g, c_0, c_1, c_2) \cdot (1, f, g, h)^{\theta'}$, and simply keep σ_2. While the correctness follows by inspection, we have several comments to make that are less obvious. First, the reason why we are no more able to modify m is due to the presence of the constant g that must be the first component of the signed vector associated to σ_1 and any adaptation σ_1'. Modifying m requires computing a one-time LHSP signature on a vector necessarily outside the span generated by (g, c_0, c_1, c_2) and $(1, f, g, h)$. Second, the signature σ_2 is unchanged during the randomization. Still, it is a signature on a fixed

[2] H must not only be second-preimage resistance as in [27] since the adversary can choose opk^* adaptively.

vector and the one-time LHSP signing algorithm is deterministic. Moreover, if we have two distinct signatures on a single vector we can solve SXDH. That means that any other adversarially generated ciphertext for opk (as the adversary might know osk) will have to share σ_2 and our randomizability holds. Third, while the tag-based simulation-sound QA-NIZK proof π can be simulated if we embed a random triple (F, G, H) into (c_0, c_1, c_2) we also have to produce a valid looking adaptation of σ_1 while we do not know osk^*. To avoid extracting osk from a costly bit-by-bit proof of knowledge in the standard model since osk consists of random scalars,[3] we would like to add $(1, F, G, H)$ in the public key and requires the ciphertext to further compute a signature σ_3 on it with osk. However, if we reveal $(1, F, G, H)$, computing $(g, c_0^*, c_1^*, c_2^*) = (g, c_0, c_1, c_2) \cdot (1, f, g, h)^{\theta^*} \cdot (1, F, G, H)^{\rho*}$ allows deriving a valid $\sigma_1^* = \sigma_1 \cdot \sigma_2^{\theta^*} \cdot \sigma_3^{\rho^*}$ but $\boldsymbol{c}^* = (c_0^*, c_1^*, c_2^*)$ will not be random even if (F, G, H) is random. Fortunately, it is actually sufficient for the traceability to use $(\mathsf{opk}, \mathsf{osk})$ to sign the shorter vectors (g, c_0, c_1), $(1, f, g)$ and $(1, F, G)$, and keep H away from the adversary's view to have a statistically random $\boldsymbol{c}^*$ in the reduction. When $(1, F, G)$ is not in $\mathrm{span}\langle(1, f, g)\rangle$, the proof π simply prevents the adversary from randomizing ciphertexts with $(1, F, G)$ without losing validity.

For technical reason, we hide σ_1 in the ciphertext and make a randomizable NIWI Groth-Sahai proof to show the randomizability and the TCCA security of the scheme. While we can adapt the σ_1 component when we randomize one of the two ciphertexts given by the adversary in the challenge phase (or in the randomization experiment), and that trace to each other, since the adversary might know osk it might infer more information about how we adapt this signature into σ_1^* if we left it in the clear.

4.4 Description

Gen(1^λ): Choose bilinear groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ of prime order $p > 2^{\mathrm{poly}(\lambda)}$ together with $g, h \leftarrow\!\$\, \mathbb{G}$ and $\hat{g}, \hat{h} \leftarrow\!\$\, \hat{\mathbb{G}}$.

1. Pick random $\alpha, \beta \leftarrow\!\$\, \mathbb{Z}_p$ and set $f = g^\alpha h^\beta$.
2. Pick $\delta \leftarrow\!\$\, \mathbb{Z}_p$ and compute $(F, G, H) = (f, g, h)^\delta$.
3. Generate a Groth-Sahai CRS $\mathsf{crs}_w = (\vec{w}_1, \vec{w}_2) \in \mathbb{G}^4$ to commit to groups elements of $\mathbb{G}$, where $\vec{w}_1 = (w_{11}, w_{12})$ and $\vec{w}_w = (w_{21}, w_{22})$ are generated in the perfect NIWI mode, i.e., $\mathsf{crs}_w \leftarrow\!\$\, \mathbb{G}^4$.
4. Define the vector $\mathbf{v} = (g, h)$ and generate 2 key pairs $(\mathsf{sk}_u, \mathsf{pk}_u)$ and $(\mathsf{sk}_u, \mathsf{pk}_u)$ for the one-time linearly homomorphic signature of Section 4.2 in order to sign vectors of dimension $n = 2$, given the common public parameters $\hat{g}, \hat{h}$. Let $\mathsf{pk}_u = \{\hat{u}_1, \hat{u}_2\}$ and $\mathsf{pk}_v = \{\hat{v}_1, \hat{v}_2\}$. Using sk_u (resp. sk_v), generate a one-time LHSP signature $\Sigma_u = (Z_u, R_u)$ (resp. $\Sigma_v = (Z_v, R_v)$) on $\mathbf{v}$. In other words, for $\mathsf{pk}_{\mathsf{lhsp}}^{\mathsf{qazk}} = \{\hat{u}_1, \hat{u}_2, \hat{v}_1, \hat{v}_2\}$, Σ_u, Σ_v are one-time LHSP signatures on the rows of the matrix

$$\mathbf{P} = \begin{pmatrix} g & h & 1 & 1 \\ 1 & 1 & g & h \end{pmatrix}.$$

[3] There is no fully structure-preserving signature schemes under SXDH and none with full randomizability (except in the generic group model [22]), which might still not be enough to be combined with a ciphertext as an SRC). And, we are not aware of any fully structure-preserving LHSP signature scheme, where the secret keys only contain source group elements.

The private key consists of $\mathsf{SK} = (\alpha, \beta)$ and the public key $\mathsf{PK} \in \mathbb{G}^{13} \times \hat{\mathbb{G}}^{6}$ is

$$\mathsf{PK} = \Big(f,\ g,\ h,\ F,\ G,\ \mathsf{crs}_w,\ \Sigma_u,\ \Sigma_v,\ \mathsf{pk}^{\mathsf{qazk}}_{\mathsf{lhsp}},\ \hat{g},\ \hat{h}\Big)\ .$$

Enc(PK, m): to encrypt a message $m \in \mathbb{G}$, first run $\mathsf{LinkGen}(\mathsf{PK})$: Generate a key pair $(\mathsf{osk}, \mathsf{opk})$ for the one-time linearly homomorphic signature of Section 4.2 from the public generators $\hat{g}, \hat{h}$ in order to sign vectors of dimension 3. Let $\mathsf{lk} = \mathsf{osk} = \{(\eta_i, \zeta_i)\}_{i=1}^{3}$ be the private key, of which the corresponding public key is $\mathsf{opk} = \{\hat{f}_i\}_{i=1}^{3}$. Then, conduct the following steps of $\mathsf{LEnc}(\mathsf{PK}, \mathsf{lk}, m)$:

1. Pick $\theta \leftarrow\!\!\$\, \mathbb{Z}_p$ and compute the CPA encryption $\mathbf{c} = (c_0, c_1, c_2)$, where $c_0 = mf^{\theta}$, $c_1 = g^{\theta}$ and $c_2 = h^{\theta}$, and keep the random coin θ.
   Next steps 2-3 are dedicated to the tracing part.
2. To allow tracing, authenticate the row space of the matrix $\mathbf{T} = \big(T_{i,j}\big)_{1 \le i,j \le 3}$

$$\mathbf{T} = \begin{pmatrix} g & c_0 & c_1 \\ 1 & f & g \\ 1 & F & G \end{pmatrix} \qquad (2)$$

   by using $\mathsf{lk} = \mathsf{osk}$. Namely, sign each row $\vec{T_i} = (T_{i,1}, T_{i,2}, T_{i,3})$ of $\mathbf{T}$ resulting in $\boldsymbol{\sigma} = (\sigma_i)_{i=1}^{3} \in \mathbb{G}^6$, where $\sigma_i = (Z_i, R_i) \in \mathbb{G}^2$.
3. To allow strong randomizability, commit to σ_1 using the Groth-Sahai CRS crs_w by computing $\boldsymbol{C}_Z = \iota(Z_1)\vec{w}_1^{z_1}\vec{w}_2^{z_2}$ and $\boldsymbol{C}_R = \iota(R_1)\vec{w}_1^{r_1}\vec{w}_2^{r_2}$. To ensure that σ_1 is a valid one-time LHSP signature on (g, c_0, c_1) compute the proof $\hat{\pi}_{\mathsf{sig}} = (\hat{P}_1, \hat{P}_2) \in \hat{\mathbb{G}}^2$ such that $\hat{P}_1 = \hat{g}^{z_1}\hat{h}^{r_1}$ and $\hat{P}_2 = \hat{g}^{z_2}\hat{h}^{r_2}$.
   Next step 4 shows the validity of $\mathbf{c}$ *associated to the tag* $\tau = H(\mathsf{opk})$.
4. Given θ and $\tau = H(\mathsf{opk})$, compute a randomizable simulation-sound proof that $(c_1, c_2) \in \mathrm{span}\langle(g, h)\rangle$. Namely, derive the LHSP signature $\pi = (\Sigma_u^{\tau}\Sigma_v)^{\theta} =: (Z_\pi, R_\pi)$ on the vector $(c_1^{\tau}, c_2^{\tau}, c_1, c_2) = ((g, h, 1, 1)^{\tau}(1, 1, g, h))^{\theta}$.

Output the ciphertext

$$\mathsf{CT} = \Big(\mathbf{c}, \boldsymbol{C}_Z, \boldsymbol{C}_R, \sigma_2, \sigma_3, \pi, \hat{\pi}_{\mathsf{sig}}, \mathsf{opk} = \{\hat{f}_i\}_{i=1}^{3}\Big) \in \mathbb{G}^{13} \times \hat{\mathbb{G}}^{5}$$

Trace(PK, CT): Parse PK and CT as above, and output opk in the obvious way.

Rand(PK, CT): If PK and CT do not parse as the outputs of Gen and Enc, abort. Otherwise, conduct the following steps:

1. Parse the CPA encryption part $\mathbf{c} = (c_0, c_1, c_2)$, pick $\theta' \leftarrow\!\!\$\, \mathbb{Z}_p$ and compute $\mathbf{c}' = \mathbf{c} \cdot (f, g, h)^{\theta'}$, so that $c_0' = c_0 f^{\theta'}$, $c_1' = c_1 g^{\theta'}$ and $c_2' = c_2 h^{\theta'}$.
2. Implicitly adapt the committed signature σ_1 of the tracing part. First, compute $\tilde{\sigma}_1 = (\tilde{Z}_1, \tilde{R}_1) = (Z_2^{\theta'}, R_2^{\theta'}) = \sigma_2^{\theta'}$, which consists of a one-time LHSP signature on $(1, f, g)^{\theta'}$ for opk. Second, adapt the commitments $\boldsymbol{C}_Z' = \boldsymbol{C}_Z \cdot \iota(\tilde{Z}_1)\vec{w}_1^{z_1'}\vec{w}_2^{z_2'}$ and $\boldsymbol{C}_R' = \boldsymbol{C}_R \cdot \iota(\tilde{R}_1)\vec{w}_1^{r_1'}\vec{w}_2^{r_2'}$, for some random scalars $z_1', z_2', r_1', r_2' \leftarrow\!\!\$\, \mathbb{Z}_p$, which should commit to the valid one-time LHSP signature $\sigma_1' = \sigma_1\sigma_2^{\theta'}$ on (g, c_0', c_1') for opk. Third, adapt the proof $\hat{\pi}_{\mathsf{sig}} = (\hat{P}_1, \hat{P}_2)$ as $\hat{\pi}_{\mathsf{sig}}' = (\hat{P}_1', \hat{P}_2')$, where $\hat{P}_1' = \hat{P}_1 \cdot \hat{g}^{z_1'}\hat{h}^{r_1'}$ and $\hat{P}_2' = \hat{P}_2 \cdot \hat{g}^{z_2'}\hat{h}^{r_2'}$.

3. Adapt the proof of the validity of the CPA ciphertext. Namely, computes $\pi' = \pi \cdot (\Sigma_u^\tau \Sigma_v)^{\theta'} = (Z_\pi (Z_u^\tau Z_v)^{\theta'}, R_\pi (R_u^\tau R_v)^{\theta'})$, where $\tau = H(\mathsf{opk})$.

Output the re-randomized ciphertext

$$\mathsf{CT} = \left(\mathbf{c}', \boldsymbol{C}'_Z, \boldsymbol{C}'_R, \sigma_2, \sigma_3, \pi', \hat{\pi}'_{\mathsf{sig}}, \mathsf{opk}\right) .$$

Ver(PK, CT): First, abort and output 0 if PK or CT does not parse properly. Second, verify the validity of the signatures σ_2 and σ_3 on the 2 last rows $\{\vec{T}_i\}_{i=2}^3$ of the matrix $\mathbf{T}$, and output 0 if it does not hold. Third, verify that:

1. The committed signature of the tracing part is valid, i.e., $\sigma_1 = (Z_1, R_1)$ is a valid one-time LHSP signature on the vector (g, c_1, c_2). To hold, the commitments $\boldsymbol{C}_Z, \boldsymbol{C}_R$ and the proof $\hat{\pi}_{\mathsf{sig}} = (\hat{P}_1, \hat{P}_2)$ must satisfy

$$E(\boldsymbol{C}_Z, \hat{g}) \cdot e(\boldsymbol{C}_R, \hat{h}) = E(\iota(g), \hat{f}_1) \cdot E(\iota(c_0), \hat{f}_2) \cdot E(\iota(c_1), \hat{f}_3) \cdot E(\vec{w}_1, \hat{P}_1) \cdot E(\vec{w}_2, \hat{P}_2)) ; \tag{3}$$

2. The proof that the CPA ciphertext is valid, i.e., $\pi = (Z_\pi, R_\pi)$ is a valid one-time LHSP signature on the vector $(c_1^\tau, c_2^\tau, c_1, c_2)$, which must satisfy

$$e(Z_\pi, \hat{g}) \cdot e(R_\pi, \hat{h}) = e(c_1, \hat{u}_1^\tau \hat{v}_1) \cdot e(c_2, \hat{u}_2^\tau \hat{v}_2), \tag{4}$$

where $\tau = H(\mathsf{opk})$.

If at least one of theses checks fails, output 0, otherwise, output 1.

Dec(SK, PK, CT): If $\mathsf{Ver}(\mathsf{PK}, \mathsf{CT}) = 0$, output $\perp$. Otherwise, given $\mathsf{SK} = (\alpha, \beta)$ and $\mathbf{c} = (c_0, c_1, c_2)$ included in CT, compute and output $m = c_0 \cdot c_1^{-\alpha} \cdot c_2^{-\beta}$.

The correctness follows by inspection.

4.5 Security

The security of our pairing-based TREnc relies solely on the SXDH assumption. We first show the verifiability of this TREnc as it eases the analysis of the traceability and the randomizability properties. The verifiability essentially relies on the unforgeability of LHSP signatures since it also implies the (simulation-) soundness of the (quasi-adaptive) proof of (subspace) membership.

Theorem 4.1. *The above TREnc is verifiable under the SXDH assumption. More precisely, for any adversary $\mathcal{A}$, we have* $\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathrm{ver}}(\lambda) = 1] \leq \varepsilon_{\mathsf{sxdh}} + 1/p$.

Proof. See Appendix D. □

Theorem 4.2. *The above TREnc Π is traceable under the SXDH assumption. More precisely, for any adversary $\mathcal{A}$, we have* $\Pr[\mathsf{Exp}_{\mathcal{A},\Pi}^{\mathrm{trace}}(\lambda) = 1] \leq 2 \cdot \varepsilon_{\mathsf{sxdh}} + 2/p$.

Proof. See Appendix D. □

Strong randomizability essentially relies on the verifiability, which shows that computationally-bounded adversary only produces (except with negligible probability) valid ciphertexts that are honest (but, possibly with biased randomness), and the perfect randomization of honest ciphertexts.

Theorem 4.3. *The above TREnc is strongly randomizable under the SXDH assumption. More precisely, for any adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, where $\mathcal{A}_2$ is possibly unbounded, we have* $\Pr[\mathsf{Exp}^{\mathrm{rand}}_{\mathcal{A}}(\lambda) = 1] \leq \varepsilon_{\mathsf{sxdh}} + 2/p$.

Proof. See Appendix D. □

Theorem 4.4. *The above TREnc is TCCA-secure under the SXDH assumption and the collision resistance of the hash function. More precisely, we have* $\Pr[\mathsf{Exp}^{\mathrm{tcca}}_{\mathcal{A},\Pi}(\lambda) = 1] \leq \frac{1}{2} + \varepsilon_{\mathsf{cr}} + 2 \cdot \varepsilon_{\mathsf{sxdh}} + \Omega(2^{-\lambda})$.

Proof. See Appendix D. □

4.6 Efficiency

This TREnc instance is reasonably efficient. In particular, in order to encrypt a message, which is typically the bottleneck in voting applications because it must run more or less transparently on low-end voter devices, we can encrypt one group element using 29 exponentiations in $\mathbb{G}$ and 10 exponentiations in $\hat{\mathbb{G}}$. This group element would make it possible to encode up to a few hundred bits in practice, depending on the chosen security parameter.

In contrast, the SRC aiming at similar applications and used in the BeleniosRF election system [14] requires 33 exponentiations in $\mathbb{G}$ and 22 exponentiations in $\hat{\mathbb{G}}$ for the (signed) encryption of only 1 bit. In general, their construction requires $11k + 22$ exponentiations in $\mathbb{G}$ and $10k + 12$ exponentiations in $\hat{\mathbb{G}}$ in order to encrypt k bits. These estimates are based on the reference code of the SRC, since the paper does not entirely specify the algorithms (especially how commitments and proofs are computed).

5 Voting scheme based on Traceable Receipt-Free Encryption

Traceable Receipt-Free Encryption schemes are particularly well suited for the design of voting systems offering receipt freeness, that is, systems in which voters cannot demonstrate how they voted to a third party.

We are now formalizing the notion of voting system (Sec. 5.1) and receipt-freeness (Sec. 5.2), using a definition closely related to the one of Chaidos et al. [14], while fixing two technical issues that it contains, then show how to build a receipt-free voting scheme from a TREnc (Sec. 5.3).

5.1 Definitions and notations

We define voting protocols in a way that largely follows the SOK from Bernhard et al. [8] and BeleniosRF [14]. In our voting protocols, we consider the following parties:

- The *voters* are participating in the election and are willing to cast a ballot representing their vote intent.

- The *election administrator* is organizing the election and is responsible for coordinating the protocol execution.
- The *ballot box manager* is gathering the ballots of the voters on a bulletin board BB and provides a public view PBB of those ballots, for verifiability.
- The *trustees* are responsible for correctly tallying the ballot box and providing a proof of the correctness of the tally. We consider a k-threshold tallying system, that is k honest trustees are required to compute the tally of the election.

These parties are standard entities in the voting literature. In some cases, we will also refer to the ballot box manager as the rerandomizing server, in order to make its receipt-freeness related role more visible. We also define a family of deterministic results functions ρ_m which given m votes, returns the result of the election for these votes. The following definition encompasses the procedures used in a voting system.

Definition 5.1 (Voting System). *A Voting System is a tuple of probabilistic polynomial-time algorithms* ($\mathsf{SetupElection}$, Vote, $\mathsf{ProcessBallot}$, $\mathsf{TraceBallot}$, Valid, Append, $\mathsf{Publish}$, $\mathsf{VerifyVote}$, Tally, $\mathsf{VerifyResult}$) *associated to a result function* $\rho_m : \mathbb{V}^m \cup \{\perp\} \rightarrow \mathbb{R}$ *where* $\mathbb{V}$ *is the set of valid votes and* $\mathbb{R}$ *is the result space such that:*

- $\mathsf{SetupElection}(1^\lambda)$*: on input security parameter* 1^λ*, generate the public and secret keys* $(\mathsf{pk}, \mathsf{sk})$ *of the election.*
- $\mathsf{Vote}(\mathsf{id}, \mathsf{v})$*: when receiving a voter* id *and a vote* v*, outputs a ballot* b *and auxiliary data* aux*. It will also be possible to call* $\mathsf{Vote}(\mathsf{id}, \mathsf{v}, \mathsf{aux})$ *in order to obtain a ballot (without auxiliary data this time) for vote* v *using* aux*. This auxiliary data will be useful to define security and enables the creation of ballots that share the same* aux*.*
- $\mathsf{ProcessBallot}(\mathsf{b})$*: on input ballot* b*, outputs an updated ballot* b'*. In our case,* b' *would be a rerandomization of* b*.*
- $\mathsf{TraceBallot}(\mathsf{b})$*: on input ballot* b*, outputs a tag* t*. The tag is the information that a voter can use to trace his ballot, using* $\mathsf{VerifyVote}$*.*
- $\mathsf{Valid}(\mathsf{BB}, \mathsf{b})$*: on input ballot box* BB *and ballot* b*, outputs* 0 *or* 1*. The algorithm outputs* 1 *if and only if the ballot is valid.*
- $\mathsf{Append}(\mathsf{BB}, \mathsf{b})$*: on input ballot box* BB *and ballot* b*, appends* $\mathsf{ProcessBallot}(\mathsf{b})$ *to* BB *if* $\mathsf{Valid}(\mathsf{BB}, b) = 1$.
- $\mathsf{Publish}(\mathsf{BB})$*: on input ballot box* BB*, outputs the public view* PBB *of* BB*, which is the one that is used to verify the election. Depending on the context, it may be used to remove some voter credentials for instance.*
- $\mathsf{VerifyVote}(\mathsf{PBB}, \mathsf{t})$*: on input public ballot box* PBB *and tag* t*, outputs* 0 *or* 1*. This algorithm is used by voters to check if their ballot has been processed and recorded properly.*
- $\mathsf{Tally}(\mathsf{BB}, \mathsf{sk})$*: on input ballot box* BB *and private key of the election* sk*, outputs the tally* r *and a proof* Π *that the tally is correct w.r.t. the result function* ρ_m*.*
- $\mathsf{VerifyResult}(\mathsf{PBB}, \mathsf{r}, \Pi)$*: on input public ballot box* PBB*, result of the tally* r *and proof of the tally* Π*, outputs* 0 *or* 1*. The algorithm outputs* 1 *if and only if* Π *is a valid proof that* r *is the election result, computed w.r.t.* ρ_m*, corresponding to the ballots on* PBB*.*

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

These algorithms are used as follows in a typical election, the election authorities first generate the election public and secret keys with $\mathsf{SetupElection}$. Then, using the public key of the election, each voter can prepare a ballot $\mathsf{b}$with the Vote algorithm and send it to the ballot box manager. The voter also keeps $\mathsf{TraceBallot}(\mathsf{b})$ in order to be able to trace its ballot on the election public bulletin board. Each time the ballot box manager receives a ballot, it checks if it is valid with the Valid algorithm. If this is the case, it runs the $\mathsf{ProcessBallot}$ algorithm on it and appends the resulting ballot to the ballot box using Append.

The ballot box manager also applies $\mathsf{Publish}$ on the ballot box in order to obtain the content that is made available on a public bulletin board PBB. Voters can check that their ballot has been correctly recorded on PBBusing $\mathsf{VerifyVote}$.

Eventually, the trustees run the Tally algorithm on the ballot box in order to compute the election result and a proof of correctness of this result. Anyone can use these, together with the content of PBB, in order to verify the election result using $\mathsf{VerifyResult}$.

This definition differs from [8] and [14] in two important ways. First, we introduce the $\mathsf{TraceBallot}$ algorithm. Such a procedure is implicit other voting system descriptions, often because voters simply check the presence on $\mathsf{PBB}$of their ballot, in which case $\mathsf{TraceBallot}$would simply be the identity function. In our case, $\mathsf{TraceBallot}$must extract the signature verification key that is generated at voting time by Vote, making this algorithm non-trivial.

The correctness guarantees of the various algorithms listed above are as usual and follow the intuitions given above. We only formalize the correctness guarantee of $\mathsf{TraceBallot}$, which is novel.

Definition 5.2 (Tracing correctness). *For every* v*,* BB*,* $(\mathsf{b}, \mathsf{aux}) \leftarrow \mathsf{Vote}(\mathsf{id}, \mathsf{v})$ *and* $\mathsf{t} \leftarrow \mathsf{TraceBallot}(\mathsf{b})$*, after* $\mathsf{Append}(\mathsf{BB}, \mathsf{b})$ *we have that* $\mathsf{VerifyVote}(\mathsf{Publish}(\mathsf{BB}), \mathsf{t}) = 1$ *with overwhelming probability.*

As a second difference, we omit the voter registration procedure, of which we make no use here: it is used in some protocols in order to obtain some forms of delegated verifiability where an extra authority is partially trusted to handle voter credentials, but this is not our focus. To make things concrete here, one can imagine that voter authentication is handled with a process similar to the one used in Helios [3], where the ballot box manager distributes credentials (e.g., passwords) to the voters and publishes the voter names next to their ballot on PBB. Voters who did not vote can then verify that there is no ballot recorded for them, and auditors can sample voters and contact them to perform similar verification steps. (We make no claim regarding the effectiveness of this process in practice – it is just here for context.)

5.2 Receipt-freeness

We adopt here a definition of receipt-freeness that is similar to the one of Chaidos et al [14], which is copied in Appendix F for completeness. Various other definitions exist, but they are either too informal for our purpose (e.g., [23]), or focus on the stronger notion of coercion resistance, in which voters need to adopt a specific counter-strategy depending on instructions of the coercer (e.g., [33,38,29,4]).

This definition requires that voters should not be able to pick a ballot, possibly from a distribution that deviates from the honest one, in such a way that no third party, by looking at the election public bulletin board, and knowing exactly how the voter's announced ballot was built, is able to decide whether that ballot was submitted by the voter rather than another ballot that could encode a vote for a different candidate. This definition also considers that the channels between the voters and the ballot box manager is private and, indeed, without the assumption that such private channels are available, achieving receipt-freeness in a verifiable election is impossible [17].

Definition 5.3 (Receipt-Freeness). *A voting system $\mathcal{V}$ has receipt-freeness if there exists 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 3, 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 $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{SetupElection}(1^\lambda)$
else $(\mathsf{pk}, \mathsf{sk}, \tau) \leftarrow \mathsf{SimSetupElection}(1^\lambda)$
$\mathsf{BB}_0 \leftarrow \perp; \mathsf{BB}_1 \leftarrow \perp$
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 $\mathsf{Valid}(\mathsf{BB}_0, \mathsf{b}_0) = 0$ or $\mathsf{Valid}(\mathsf{BB}_1, \mathsf{b}_1) = 0$
then return $\perp$
else $\mathsf{Append}(\mathsf{BB}_0, \mathsf{b}_0); \mathsf{Append}(\mathsf{BB}_1, \mathsf{b}_1)$
```

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

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

Fig. 3. 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.

In this game, parameterized by a bit β, the adversary has access to the following oracles:

- $\mathcal{O}\mathsf{init}$: initializes the voting system. It generates the public and privates keys of election and returns the public key to the adversary. When $\beta = 1$, a simulated setup may be performed, depending on the computational model, which will offer some trapdoor information that may be needed to produced a simulated tally correctness proof for instance. Eventually, two empty ballot boxes are created: the real ballot box BB_0 that will be tallied and the fake ballot box BB_1. Both boxes will be populated during the game, but the adversary will only see $\mathsf{Publish}(\mathsf{BB}_\beta)$.
- $\mathcal{O}\mathsf{receiptLR}(\mathsf{b}_0, \mathsf{b}_1)$: Lets the adversary cast ballots b_0 in the real ballot box BB_0 and b_1 in the fake ballot box BB_1, as long as both ballots are valid and have the same trace. This oracle is the central one for receipt-freeness.
- $\mathcal{O}\mathsf{board}$: Returns $\mathsf{Publish}(\mathsf{BB}_\beta)$, which represents its view of the public bulletin board.

– $\mathcal{O}\mathsf{tally}$: Returns the result of the election based on the ballots on BB_0, as well as a proof of correctness of the tally. If $\beta = 1$, this proof is simulated w.r.t. the content derived from BB_1.

Several observations can be made about this game. First, and as expected, it can be seen that the ballot box manager is considered to behave honestly. A dishonest ballot box manager could simply replace $\mathsf{ProcessBallot}$ and $\mathsf{Publish}$ with the identity function, which would make the game trivial to win, independently of any cryptographic operation. The Tally operation is also performed honestly: dishonest talliers could decrypt all the ballots individually, which would again make the game trivial to win. In practice, this assumption can be mitigated by using a distributed decryption process, which is always possible using MPC but can typically be done more efficiently.

Second, this game prompts for the introduction of an extra correctness requirement on the definition of Vote and $\mathsf{TraceBallot}$, in order to make sure that ballots that encode different votes and have the same tag can be computed.

Definition 5.4 (Ballot traceability for receipt freeness). *For every public key* pk *in the range of* $\mathsf{SetupElection}$*, the ballots produced for every pair of voting choices* $(\mathsf{v}_0, \mathsf{v}_1)$ *with the same auxiliary data trace to the same tag. That is, for* $\mathsf{b}_0, \mathsf{aux} \leftarrow\$ \mathsf{Vote}(\mathsf{id}, \mathsf{v}_0)$*,* $\mathsf{b}_1 \leftarrow\$ \mathsf{Vote}(\mathsf{id}, \mathsf{v}_1, \mathsf{aux})$*, we have* $\mathsf{TraceBallot}(\mathsf{b}_0) = \mathsf{TraceBallot}(\mathsf{b}_1)$*.*

Without this extra constraint, we could imagine a $\mathsf{TraceBallot}$ algorithm which returns a tag depending on the vote inside the ballot. For example if $\mathsf{TraceBallot}(\mathsf{b}) = \mathsf{b}$ as we discussed earlier, then $\mathcal{O}\mathsf{receipt}(\mathsf{b}_0, \mathsf{b}_1)$ does nothing except if the two ballots are identical and the adversary can never win the receipt-freeness game. It is thus natural to require that a tag returned by $\mathsf{TraceBallot}$ can be reached with any possible voting choice.

The related constraint that $\mathsf{TraceBallot}(\mathsf{b}_0) = \mathsf{TraceBallot}(\mathsf{b}_1)$ is actually missing from Chaidos' definition[14], and makes their game trivial to win in most natural case, including with their own protocol: an adversary could simply submit two ballots that have different traces (or are signed with different keys in the wording of their paper), and immediately identify which bulletin board he sees.

Compared to Chaidos' definition, we also removed the $\mathcal{O}\mathsf{corrupt}$ oracle, as we simply assume that all the voters are under adversarial control. We also omitted their $\mathcal{O}\mathsf{cast}$ and $\mathcal{O}\mathsf{voteLR}$ oracles, because $\mathcal{O}\mathsf{receiptLR}$ subsumes them.

5.3 Voting scheme

We now explain how a generic voting system can be built from a TREnc. The protocol in itself is of little interest: it essentially follows previous proposals [10,14]. Its central interest is that defining it from a TREnc makes the proof of its receipt-freeness almost immediate, and independent of any specific TREnc instance.

A detailed pseudocode description is proposed in Figure 5.3. The protocol executes as follow. The election authorities set-up the election in the following way. They create an empty bulletin board and create the pair of public and private keys $(\mathsf{pk}, \mathsf{sk})$ of the election by running the Gen algorithm of the TREnc scheme with the desired security parameter. pk is distributed to every party taking part to the election and sk is given to the tallier. Note that sk can be generated in a distributed way, so that decryption requires the contribution of multiple trustees – our TREnc constructions are compatible with standard threshold key generation protocols for discrete log based cryptosystems, which can be used as usual since the decryption of a ciphertext is independent from the link key and the Trace algorithm [18,21]. We consider a unique tallier in the following.

When a user wants to cast a vote v, they first generate a link key lk by running the $\mathsf{LGen}(\mathsf{pk})$ algorithm, then encrypts the vote with the $\mathsf{LEnc}(\mathsf{pk}, \mathsf{lk}, \mathsf{v})$ algorithm in order to obtain a ballot b, while aux is defined as lk. The voter then sends the encrypted ballot to the ballot box manager of the voting system. It is utterly important that the user erases the link key as soon as possible, as the integrity of their vote may rely on the secrecy of this key. The voter will however store $\mathsf{TraceBallot}(\mathsf{b}) = \mathsf{Trace}(\mathsf{b})$ in order to verify that a ballot with the correct trace eventually appears on the public bulletin board.

When the ballot box manager receives a new ballot b, he verifies the validity of the ballot by checking that $\mathsf{Ver}(\mathsf{b})$ succeeds and that no ballot with the same trace was recorded before. Invalid ballots are dropped and valid ones are going through $\mathsf{Append}(\mathsf{BB}, \mathsf{b})$, which runs $\mathsf{ProcessBallot}(\mathsf{b}) = \mathsf{Rand}(\mathsf{pk}, \mathsf{b})$ and appends the result to BB.

The user can verify that their vote is on the bulletin board by checking with the $\mathsf{TraceBallot}$ algorithm if any entry in the public bulletin board has the same trace as the one they recorded when they produced their ballot. The traceability property of the TREnc then guarantees that nobody (including the rerandomizing server and the election authorities who hold the decryption key) could have forged another valid ciphertext of another vote linked to this ballot with non-negligible probability.

Once every voter has cast a vote, the tallier can gather the ballots on the bulletin board and compute the result of the election r, as well as a proof of correctness Π. The exact details of this process will depend on the ballot format and the result function ρ_m that the voting protocol requires. One standard way of performing this operation would be to process all the published ballots through a verifiable mixnet: our TREnc ciphertexts are compatible with various standard options that operate on votes encrypted as vectors of group elements, including the Verificatum mixnet [40].

5.4 Security of the voting scheme

This voting scheme has receipt freeness, as claimed in the following theorem.

Theorem 5.5. *If the TREnc used in the voting scheme is TCCA secure and verifiable and if the proof system used to prove the correctness of the tally is zero-knowledge, then the voting scheme has receipt freeness. More precisely, if the advantage of any adversary at distinguishing a simulator from an honest prover of the proof system is bounded by ε_{ZK} and if the advantage of any adversary at the* TCCA *experiment is bounded by a negligible function ε_{TCCA}, then every adversary at the receipt freeness game making q_r $\mathcal{O}\mathsf{receiptLR}$ requests has an advantage bounded by $\varepsilon_{ZK} + q_r\varepsilon_{TCCA}$.*

Proof. The proof uses two different games, where the first one is the receipt-freeness game. In each of those games, we pick a random bit β corresponding to either the real ballot box ($\beta = 0$) or the fake ballot box ($\beta = 1$). The adversary is expected to guess in which case it is and to output a

```
SetupElection (λ)
-----------------
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$
return $\mathsf{pk}$
```

```
Vote(id, v[, aux])
------------------
if aux is specified then $\mathsf{lk} \leftarrow \mathsf{aux}$
else $\mathsf{lk} \leftarrow\!\$\ \mathsf{LGen}(\mathsf{pk})$
$\mathsf{b} \leftarrow \mathsf{LEnc}(\mathsf{pk}, \mathsf{lk}, \mathsf{v})$
if aux is not specified then return $\mathsf{b}$
else return $\mathsf{b}, \mathsf{lk}$
```

```
TraceBallot(b)
--------------
return $\mathsf{Trace}(\mathsf{b})$
```

```
Valid(BB, b)
------------
if $\mathsf{Ver}(\mathsf{b}) \wedge$
$\forall \mathsf{b}' \in \mathsf{BB}, \mathsf{TraceBallot}(\mathsf{b}') \neq \mathsf{TraceBallot}(\mathsf{b})$
then return $1$ else return $0$
```

```
ProcessBallot(b)
----------------
return $\mathsf{Rand}(\mathsf{pk}, \mathsf{b})$
```

```
VerifyVote(PBB, t)
------------------
if $\exists b \in \mathsf{PBB} : \mathsf{Valid}(b) \wedge \mathsf{t} == \mathsf{TraceBallot}(\mathsf{b})$
then return $1$ else return $0$
```

Fig. 4. Instantiation of our voting scheme from a generic TREnc scheme. Publish is simply the identity function. Tally and VerifyResult are instantiated via standard techniques, depending on the result function (homomorphic tallying, verifiable mixnet, . . .).

bit β'. We note S_i the event that $\beta = \beta'$ in the i-th game. We show that S_0, the probability of an adversary to win the receipt freeness game, is negligibly close in λ to $\frac{1}{2}$.

$\mathsf{Game}_0(\lambda)$: We define $\mathsf{Game}_0(\lambda)$ as the original receipt freeness experiment and $\mathcal{A}$ as a PPT adversary for the game. We set SimSetup as the simulation trapdoor for the proof systems used in the tally and $\mathsf{SimProof}(\mathsf{BB}, \mathsf{r})$ as an algorithm simulating fake proofs of the decryption of the ciphertexts in BB into the plaintexts listed in r. By definition, $\mathcal{A}$ wins the game with probability $Pr[S_0]$.

$\mathsf{Game}_1(\lambda)$: If $\beta = 0$, we generate the keys of the election with SimSetup instead of the honest Setup algorithm. We also give a simulated proof of the tally as in the case $\beta = 1$.
$\mathsf{Game}_0(\lambda) \rightarrow \mathsf{Game}_1(\lambda)$: Since the proof system of the tally is zero-knowledge, $|Pr[S_0] - Pr[S_1]| \leq \varepsilon_{ZK}$.

In the second game, the only difference between $\beta = 0$ and $\beta = 1$ are now in the $\mathcal{O}\mathsf{receiptLR}$ oracle. We can reduce the q-TCCA experiment (the q challenge variant of the TCCA game, which is proven to be equivalent to TCCA security up to a factor q in Appendix B) to $\mathsf{Game}_1(\lambda)$ in the following way. We build an adversary against the q-TCCA challenger by instantiating the voting system and simulating an efficient adversary for $\mathsf{Game}_1(\lambda)$. Each time we are asked to append ballots to the bulletin board from a $\mathcal{O}\mathsf{receiptLR}$ oracle call, we ask the challenger to decrypt them. Then, we give both ballots as a challenge to the challenger and receive a randomized ballot that we append to our bulletin board. There are q_r such requests. After all the requests, we can compute the result of the election as we have the plaintext of every ballot in BB_0. Moreover, we can use SimProof to simulate a proof that our bulletin board has been correctly tallied. Hence, this adversary wins the q-TCCA game with the same probability as the simulated adversary wins the second game and

$Pr[S_1] \leq q_r \varepsilon_{TCCA}$. We conclude that the probability that the adversary wins the receipt freeness experiment, $Pr[S_0]$, is bounded by $\varepsilon_{ZK} + q_r \varepsilon_{TCCA}$. □

It is immediate that our voting scheme also satisfies ballot traceability (Def. 5.4), thanks to the link traceability of the TREnc (Def. 2.2).

This demonstrates the receipt-freeness of our protocol against an adversary who sees the public bulletin board, and assuming a honest ballot box manager. Our protocol also offers privacy against a malicious ballot box manager, as demonstrated in Appendix G. As can be expected, proving privacy against such an adversary requires taking advantage of the strong-randomization property of the TREnc, which was not necessary for receipt-freeness.

It is also important to note that the notions of receipt-freeness and ballot privacy only make sense when applied to voting protocols that satisfy some extra correctness requirements (see Bernhard et al. [8] for instance) – a pathological Valid process that would just drop all but one ballot could result in this ballot been tallied alone, which could satisfy the definition or receipt-freeness but would obviously be problematic from a privacy point of view. The notions of strong consistence, correctness, and validity, defined in Appendix G, address these questions.

We do not detail the verifiability of our voting system, which would require to introduce a substantial machinery. We outline how this could work here:

- Individual verifiability requires that a voter who successfully completes the VerifyVote verification steps can be convinced that his vote is properly recorded. If the voter's voting client is honest, this follows from the traceability property of the TREnc and the single use of lk, which guarantee that any ballot with the same trace as the ballot submitted by the voter would decrypt to the right vote. Detecting a malicious voting client that may encrypt a vote different of the one chosen by the voter is more tricky. One option would be to consider a variation on the Benaloh challenge, in which voters would have the option to decide to spoil a ballot that has been posted on the public bulletin board, and either ask for its decryption, or for the randomness used both during the Vote process and during ProcessBallot. Any newly created ballot would need to be generated using a fresh lk.
- Eligibility verifiability could proceed by adding the voter's name next to each ballot on the public bulletin board, and let auditors check whether these are legitimate voters. Weaker but more convenient options include relying on a trusted authentication server and/or on a PKI.
- Universal verifiability, which guarantees that the tally is computed correctly, would result from the tallying process, e.g., from a verifiable mix-net.

Acknowledgments. Henri Devillez is a research fellow of the Belgian FRIA. Thomas Peters is research associate of the Belgian Fund for Scientific Research (F.R.S.-FNRS). This work has been funded in parts by the F.R.S.–FNRS SeVote project.

References

1. 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)
2. Abe, M., Haralambiev, K., Ohkubo, M.: Signing on elements in bilinear groups for modular protocol design. Cryptology ePrint Archive, Report 2010/133 (2010), https://ia.cr/2010/133
3. Adida, B., de Marneffe, O., Pereira, O., Quisquater, J.: Electing a university president using open-audit voting: Analysis of real-world use of helios. In: 2009 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections, EVT/WOTE '09. USENIX Association (2009)
4. Alwen, J., Ostrovsky, R., Zhou, H., Zikas, V.: Incoercible multi-party computation and universally composable receipt-free voting. In: Advances in Cryptology - CRYPTO 2015. LNCS, vol. 9216, pp. 763–780. Springer (2015)
5. Bauer, B., Fuchsbauer, G.: Efficient signatures on randomizable ciphertexts. In: Security and Cryptography for Networks. pp. 359–381. Springer International Publishing (2020)
6. Benaloh, J.: Simple verifiable elections. In: 2006 USENIX/ACCURATE Electronic Voting Technology Workshop, EVT'06. USENIX Association (2006)
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: 2015 IEEE Symposium on Security and Privacy. pp. 499–516 (2015)
9. Bernhard, D., Pereira, O., Warinschi, B.: How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to helios. In: Advances in Cryptology - ASIACRYPT 2012. LNCS, vol. 7658, pp. 626–643. Springer (2012)
10. Blazy, O., Fuchsbauer, G., Pointcheval, D., Vergnaud, D.: Signatures on randomizable ciphertexts. In: Public Key Cryptography – PKC 2011. pp. 403–422. Springer (2011)
11. Boneh, D., Sahai, A., Waters, B.: Functional encryption: Definitions and challenges. In: Theory of Cryptography. pp. 253–273. Springer (2011)
12. Canetti, R., Gennaro, R.: Incoercible multiparty computation (extended abstract). In: 37th Annual Symposium on Foundations of Computer Science, FOCS '96. pp. 504–513. IEEE Computer Society (1996)
13. Canetti, R., Krawczyk, H., Nielsen, J.B.: Relaxing chosen-ciphertext security. In: Advances in Cryptology - CRYPTO 2003. pp. 565–582. Springer (2003)
14. Chaidos, P., Cortier, V., Fuchsbauer, G., Galindo, D.: Beleniosrf: A non-interactive receipt-free electronic voting scheme. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS). pp. 1614–1625 (2016)
15. Chase, M., Kohlweiss, M., Lysyanskaya, A., Meiklejohn, S.: Malleable proof systems and applications. In: Advances in Cryptology – EUROCRYPT 2012. pp. 281–300. Springer (2012)
16. Chaum, D.: Surevote: technical overview. In: Proceedings of the Workshop onTrustworthy Elections (WOTE 2001) (2001)
17. Chevallier-Mames, B., Fouque, P., Pointcheval, D., Stern, J., Traoré, J.: On some incompatible properties of voting schemes. In: Towards Trustworthy Elections, New Directions in Electronic Voting. LNCS, vol. 6000, pp. 191–199. Springer (2010)
18. Cortier, V., Galindo, D., Glondu, S., Izabachène, M.: Distributed elgamal à la pedersen: Application to helios. In: Proceedings of the 12th annual ACM Workshop on Privacy in the Electronic Society, WPES 2013. pp. 131–142. ACM (2013)
19. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. In: Advances in Cryptology - EUROCRYPT '97. LNCS, vol. 1233, pp. 103–118. Springer (1997)
20. Gennaro, R.: Achieving independence efficiently and securely. In: Proceedings of the Fourteenth Annual ACM Symposium on Principles of Distributed Computing. pp. 130–136. ACM (1995)
21. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. J. Cryptol. **20**(1), 51–83 (2007)
22. Groth, J.: Efficient fully structure-preserving signatures for large messages. In: Advances in Cryptology – ASIACRYPT 2015. pp. 239–259. Springer (2015)

23. Hirt, M., Sako, K.: Efficient receipt-free voting based on homomorphic encryption. In: Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques Proceeding. LNCS, vol. 1807, pp. 539–556. Springer (2000)
24. Horwitz, J., Lynn, B.: Toward hierarchical identity-based encryption. In: Advances in Cryptology — EUROCRYPT 2002. pp. 466–481. Springer (2002)
25. Jutla, C.S., Roy, A.: Shorter quasi-adaptive nizk proofs for linear subspaces. In: Advances in Cryptology - ASIACRYPT 2013. pp. 1–20. Springer (2013)
26. Kiayias, A., Zacharias, T., Zhang, B.: End-to-end verifiable elections in the standard model. In: Advances in Cryptology - EUROCRYPT 2015. LNCS, vol. 9057, pp. 468–498. Springer (2015)
27. Kiltz, E.: Chosen-ciphertext security from tag-based encryption. In: Theory of Cryptography. pp. 581–600. Springer (2006)
28. Kiltz, E., Wee, H.: Quasi-adaptive nizk for linear subspaces revisited. Cryptology ePrint Archive, Report 2015/216 (2015), https://ia.cr/2015/216
29. Küsters, R., Truderung, T., Vogt, A.: A game-based definition of coercion-resistance and its applications. In: Proceedings of the 23rd IEEE Computer Security Foundations Symposium, CSF 2010. pp. 122–136. IEEE Computer Society (2010)
30. 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)
31. Libert, B., Peters, T., Joye, M., Yung, M.: Non-malleability from malleability: Simulation-sound quasi-adaptive nizk proofs and CCA2-secure encryption from homomorphic signatures. In: Advances in Cryptology – EUROCRYPT 2014. pp. 514–532. Springer (2014)
32. MacKenzie, P.D., Reiter, M.K., Yang, K.: Alternatives to non-malleability: Definitions, constructions, and applications (extended abstract). In: Theory of Cryptography, TCC. LNCS, vol. 2951, pp. 171–190. Springer (2004)
33. Moran, T., Naor, M.: Receipt-Free Universally-Verifiable Voting with Everlasting Privacy. In: Advances in Cryptology - CRYPTO 2006. pp. 373–392. LNCS, Springer (2006)
34. Prabhakaran, M., Rosulek, M.: Homomorphic encryption with cca security. In: Automata, Languages and Programming. pp. 667–678. Springer (2008)
35. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Advances in Cryptology – EUROCRYPT 2005. pp. 457–473. Springer (2005)
36. Sako, K., Kilian, J.: Receipt-free mix-type voting scheme - A practical solution to the implementation of a voting booth. In: Advances in Cryptology - EUROCRYPT '95. LNCS, vol. 921, pp. 393–403. Springer (1995)
37. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Advances in Cryptology. pp. 47–53. Springer (1985)
38. Unruh, D., Müller-Quade, J.: Universally composable incoercibility. In: Proceedings of the 30th Annual Conference on Advances in Cryptology. p. 411–428. CRYPTO'10, Springer-Verlag (2010)
39. Wikström, D.: Simplified submission of inputs to protocols. In: Security and Cryptography for Networks, 6th International Conference, SCN 2008, Proceedings. LNCS, vol. 5229, pp. 293–308. Springer (2008)
40. Wikström, D.: A commitment-consistent proof of a shuffle. In: Information Security and Privacy, 14th Australasian Conference, ACISP 2009 Proceedings. LNCS, vol. 5594, pp. 407–421. Springer (2009)

A Related security definitions

A.1 *RCCA* security

We recall the *RCCA* security definition introduced by Canetti et al.[13]:

Definition A.1. *An encryption scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is RCCA-secure if for every* PPT *adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *the experiment* $\mathsf{Exp}^{\mathrm{rcca}}_{\mathcal{A}}(\lambda)$ *defined in Figure 5 returns* 1 *with a probability negligibly close in* λ *to* $\frac{1}{2}$.

```
$\mathsf{Exp}^{\mathrm{rcca}}_{\mathcal{A}}(\lambda)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{Gen}(1^\lambda)$
$(m_0, m_1, \mathsf{st}) \leftarrow\$ \mathcal{A}_1^{\mathsf{Dec}(\mathsf{sk},\cdot)}(\mathsf{pk})$
$b \leftarrow\$ \{0,1\}$
$c^\star \leftarrow\$ \mathsf{Enc}(\mathsf{pk}, m_b)$
$b' \leftarrow\$ \mathcal{A}_2^{\mathsf{Dec}^{post}(\mathsf{sk},\cdot)}(c^\star, \mathsf{st})$
return $b = b'$
```

Fig. 5. *RCCA* experiment. In the experiment, $\mathcal{A}_2$ has access to a decryption oracle $\mathsf{Dec}^{post}(\cdot)$ which, on input c, returns $\mathsf{Dec}(\mathsf{sk}, c)$ if $\mathsf{Dec}(\mathsf{sk}, c) \notin \{m_0, m_1\}$ and test otherwise.

A.2 pd-RCCA

We recall the pd-RCCA security definition introduced by Canetti et al. [13]:

Definition A.2. *Let* $S = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be an encryption scheme.*

1. *We say that a family of binary relations* $\equiv_{pk}$ *(indexed over the public keys of* S*) on ciphertext pairs is a* compatible *relation for* S *if for all key-pairs* $(\mathsf{pk}, \mathsf{sk})$ *of* S *we have:*
   - *For any two ciphertexts* c, c', *if* $c \equiv_{pk} c'$ *then* $\mathsf{Dec}(\mathsf{sk}, c) = \mathsf{Dec}(\mathsf{sk}, c')$ *with overwhelming probability.*
   - *For any plaintext* m *in the domain of* S, *if* c *and* c' *are two ciphertexts obtained as independent encryptions of* m, *then* $c \equiv_{pk} c'$ *with negligible probability.*
2. *We say that a relation family as above is* publicly computable *if for all key pairs* $(\mathsf{pk}, \mathsf{sk})$ *and all pair of ciphertexts* (c, c'), *it can be determined whether* $c \equiv_{pk} c'$ *using a* PPT *algorithm taking as inputs* (pk, c, c').
3. *We say that* S *is* publicly-detectable replayable-CCA *(pd-RCCA in short) if there exists a compatible and publicly computable relation family* $\equiv_{\mathsf{pk}}$ *such that* S *is secure according to the traditional definition of CCA security with the following modification. In the post-challenge decryption requests, the oracle returns* test *for the decryption of* c *if and only if* $c \equiv_{pk} c^\star$ *where* $c^\star$ *is the challenge ciphertext.*

B Relation between the security definitions

B.1 pd*-RCCA and extractability implies TCCA

Theorem B.1. *If a TREnc scheme T is extractable, strongly randomizable and* pd*-RCCA*-secure, then it is* TCCA *secure. More precisely, if the advantages of any PPT adversary at strong randomization and* pd*-RCCA *experiment are respectively bounded by ε_{SR} and ε, then for any* PPT *adversary $\mathcal{A}$, we have $Pr[\mathsf{Exp}^{\mathrm{tcca}}_{\mathcal{A},T}(\lambda) = 1] \leq \frac{1}{2} + \varepsilon_{SR} + \varepsilon$.*

Proof. The proof uses a sequence of games starting with the TCCA game and ending with the pd*-RCCA game. In each of those games, we pick a random bit b and the adversary is expected to output a bit b'. We note S_i the event that $b = b'$ in the i-th game, that is the event that the adversary correctly guesses which message was encrypted in the challenge ciphertext. We show that S_0, the probability of an adversary to win the TCCA game, is bounded by $\frac{1}{2} + \varepsilon_{SR} + \varepsilon$.

$\mathsf{Game}_0(\lambda)$**:** Let $\mathcal{A}$ be a PPT adversary. We define $\mathsf{Game}_0(\lambda)$ as the original TCCA game $\mathsf{Exp}^{\mathrm{tcca}}_{\mathcal{A},T}(\lambda)$ as defined in Definition 2.4. By definition, $\mathcal{A}$ wins the game with probability $Pr[S_0]$.

$\mathsf{Game}_1(\lambda)$**:** We transform $\mathsf{Game}_0(\lambda)$ into $\mathsf{Game}_1(\lambda)$ by replacing the honest key generation algorithm Gen by the trapdoor algorithm $\mathsf{TrapGen}$ of Definition 3.1.

$\mathsf{Game}_0(\lambda) \rightarrow \mathsf{Game}_1(\lambda)$: From the trapdoor indistinguishability of Definition 3.1, the distribution of the pairs of keys $(\mathsf{pk}, \mathsf{sk})$ outputted by Gen and $\mathsf{TrapGen}$ are identical. Since the trapdoor tk is not used yet, the two games are thus statistically indistinguishable and $Pr[S_0] = Pr[S_1]$.

$\mathsf{Game}_2(\lambda)$**:** We transform $\mathsf{Game}_1(\lambda)$ into $\mathsf{Game}_2(\lambda)$ in the following way. After receiving the two challenge ciphertexts, we run $\mathsf{LExtr}(\mathsf{tk}, c_0)$ and $\mathsf{LExtr}(\mathsf{tk}, c_1)$ to extract the link keys lk_0 and lk_1. If $\mathsf{lk}_0 \neq \mathsf{lk}_1$, we return a random bit. Otherwise, we continue the execution of the experiment.

$\mathsf{Game}_1(\lambda) \rightarrow \mathsf{Game}_2(\lambda)$: Because the scheme has unique extraction, we always have $\mathsf{lk}_0 = \mathsf{lk}_1$. Hence, $Pr[S_1] = Pr[S_2]$.

$\mathsf{Game}_3(\lambda)$**:** We transform $\mathsf{Game}_2(\lambda)$ into $\mathsf{Game}_3(\lambda)$ by modifying the way the challenge ciphertext is created. Given c_b, we decrypt the plaintext $m = \mathsf{Dec}(\mathsf{sk}, c_b)$ and compute the challenge ciphertext $c^\star$ as $\mathsf{LEnc}(\mathsf{pk}, \mathsf{lk}_b, m)$.

$\mathsf{Game}_2(\lambda) \rightarrow \mathsf{Game}_3(\lambda)$: The initial challenge ciphertext is computed as in the left part of the strong randomization equation of Definition 2.6 while the new challenge ciphertext is computed exactly as in the right part. The unique extraction property guarantees that both c_b and $c^\star$ are in the range of $\mathsf{LEnc}(\mathsf{pk}, \mathsf{lk}_b, m)$. Since the TREnc is strongly randomizable, the distributions of these two ciphertexts are indistinguishable and $|Pr[S_2] - Pr[S_3]| \leq \varepsilon_{SR}$.

$\mathsf{Game}_4(\lambda)$**:** We transform $\mathsf{Game}_3(\lambda)$ into $\mathsf{Game}_4(\lambda)$ by modifying the way the challenge ciphertext is created. In this game, we do not generate the pair of public/secret keys anymore and instantiate a pd*-RCCA experiment instead. Each time we decrypt a ciphertext or extract a link key in $\mathsf{Game}_3(\lambda)$, we request the pd*-RCCA challenger. Instead of computing ourselves the challenge, we send $(m_0, m_1, \mathsf{lk}_0)$ to the challenger (Note that $\mathsf{lk}_0 = \mathsf{lk}_1$) and receive a ciphertext $c^\star$ that we send to the adversary.

$\mathsf{Game}_3(\lambda) \to \mathsf{Game}_4(\lambda)$: In this new game, the challenge ciphertext and the post-challenge decryption requests are computed exactly as in the previous game. Hence, $Pr[S_3] = Pr[S_4]$

Finally, $\mathsf{Game}_4(\lambda)$ is equivalent to the pd*-RCCA experiment. Since the scheme is pd*-RCCA secure, we conclude that $Pr[4] \leq \frac{1}{2} + \varepsilon$ and thus that the scheme is TCCA secure. $\square$

B.2 Link between TCCA and *RCCA* security

Theorem B.2. *If a TREnc is traceable and* TCCA *secure, then it is RCCA secure. More precisely, if*

- *The TREnc is traceable. That is, the advantage of any traceability adversary is bounded by a negligible function* ε_{TR}*.*
- *The TREnc is randomizable. That is, the advantage at distinguishing the randomization of an honest ciphertext from a fresh re-encryption is bounded by a negligible function* ε_{RAND}*.*
- *The TREnc is* TCCA *secure. That is, the advantage of any* TCCA *adversary is bounded by a negligible function* ε_{TCCA}

then the advantage of any RCCA adversary is bounded by the negligible function $\varepsilon_{TR} + \varepsilon_{RAND} + \varepsilon_{TCCA}$*.*

Proof. The proof uses a sequence of games starting with the *RCCA* game and ending with the TCCA game. In each of those games, we pick a random bit b and the adversary is expected to output a bit b'. We note S_i the event that $b = b'$ in the i-th game, that is the event that the adversary correctly guesses which message was encrypted in the challenge ciphertext. We show that S_0, the advantage of an adversary at the *RCCA* game is bounded by $\frac{1}{2} + \varepsilon_{TR} + \varepsilon_{RAND} + \varepsilon_{TCCA}$, where ε_{TR}, ε_{RAND} and ε_{TCCA} are respectively the advantages of some efficient adversaries playing the traceability, randomizability and TCCA experiments.

$\mathsf{Game}_0(\lambda)$**:** Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a PPT adversary. We define $\mathsf{Game}_0(\lambda)$ as the original *RCCA* game $\mathsf{Exp}^{\mathrm{rcca}}_{\mathcal{A}}(\lambda)$ of Definition A.2. By definition, $\mathcal{A}$ wins the game with probability $Pr[S_0]$.

$\mathsf{Game}_1(\lambda)$**:** We transform $\mathsf{Game}_0(\lambda)$ into $\mathsf{Game}_1(\lambda)$ by modifying the post-challenge decryption oracle in the following way. We replace $\mathsf{Dec}^{post}(\mathsf{sk}, c)$ with the oracle $\mathsf{Dec}^{postR}(\mathsf{sk}, c)$ that returns test if $\mathsf{Dec}(\mathsf{sk}, c) \in \{m_0, m_1\}$ or if $\mathsf{Trace}(c) = \mathsf{Trace}(c^\star)$. It returns $\mathsf{Dec}(\mathsf{sk}, c)$ otherwise.

$\mathsf{Game}_0(\lambda) \to \mathsf{Game}_1(\lambda)$: Let F be the event in $\mathsf{Game}_1(\lambda)$ that at least one of the post-challenge decryption request is such that $\mathsf{Trace}(c) = \mathsf{Trace}(c^\star)$ and $\mathsf{Dec}(c) \notin \{m_0, m_1\}$. It is clear that the two games are identical unless F occurs. Thus, $|Pr[S_0] - Pr[S_1]| \leq Pr[F]$. We claim that $Pr[F] \leq \epsilon_{TR}$, where ϵ_{TR} is the advantage at the traceability game of the following adversary $\mathcal{B}$. We build $\mathcal{B}$ as a PPT algorithm simulating an adversary for $\mathsf{Game}_1(\lambda)$ and receiving the pair of public-secret keys from the $\mathsf{Exp}_{\mathcal{B}}^{\text{trace}}(\lambda)$ challenger. $\mathcal{B}$ forwards the decryption requests of the simulated adversary to its challenger and chooses the challenge bit b. Once it receives the challenge plaintexts m_0 and m_1, it sends m_b to its challenger and forwards the resulting ciphertext to the simulated adversary. For each of the following ciphertext, $\mathcal{B}$ checks if the ciphertext is in relation with the challenge ciphertext and if it is decrypted into a message different than m_b. In that case, it sends it to the traceability challenger and wins the game. Hence $\mathcal{B}$ has an advantage that is at least $Pr[F]$.

$\mathsf{Game}_2(\lambda)$: We transform $\mathsf{Game}_1(\lambda)$ into $\mathsf{Game}_2(\lambda)$ by modifying how the challenge is built in the following way. After encrypting the challenge ciphertext, we also rerandomize it. That is, we compute $c' \leftarrow\!\$\ \mathsf{Enc}(\mathsf{pk}, m_b), c^\star \leftarrow\!\$\ \mathsf{Rand}(\mathsf{pk}, c')$ and return $c^\star$.

$\mathsf{Game}_1(\lambda) \to \mathsf{Game}_2(\lambda)$: Because the TREnc is randomizable, the randomization of a ciphertext is statistically indistinguishable from a fresh encryption and those two games are indistinguishable.
Thus $|Pr[S_1] - Pr[S_2]| \leq \varepsilon_{RAND}$.

$\mathsf{Game}_3(\lambda)$: We transform $\mathsf{Game}_2(\lambda)$ into $\mathsf{Game}_3(\lambda)$ by instantiating a TCCA game. Now, we only receive the public key pk from our TCCA challenger but we use the challenger to simulate the decryption requests. In the challenge phase, we compute a ciphertext for both messages such that they are in relation (by using $\mathsf{lk} \leftarrow\!\$\ \mathsf{LGen}(\mathsf{pk}), c_b \leftarrow\!\$\ \mathsf{LEnc}(\mathsf{pk}, \mathsf{lk}, m_b)$) and send them to the challenger. We receive a ciphertext $c^\star$ that is the randomization of either c_0 or c_1. We return it as the challenge ciphertext. Finally, we receive the adversary's guess b and send it to the oracle as our own guess. It is clear that $Pr[S_3]$ is equal to the probability of winning the rerandomization indistinguishability game.

$\mathsf{Game}_2(\lambda) \to \mathsf{Game}_3(\lambda)$: Even though we do not compute ourselves the answers of the requests anymore in $\mathsf{Game}_3(\lambda)$, they are computed exactly as in $\mathsf{Game}_2(\lambda)$ and thus $Pr[S_2] = Pr[S_3]$.

Also, this final game is identical to the TCCA security game, hence $Pr[S_3] \leq \varepsilon_{TCCA}$. From this sequence of games, we conclude that the probability of winning of any $RCCA$ adversary is bounded by a negligible function $\varepsilon_{TR} + \varepsilon_{RAND} + \varepsilon_{TCCA}$. □

B.3 TCCA implies q-TCCA

We define the q-TCCA experiment as the TCCA experiment with the following modification: instead of submitting only one pair of challenge ciphertexts, the adversary can ask for the randomization of q such pairs.

Theorem B.3. *If a TREnc is* TCCA *secure then it is* q-TCCA *as well. More precisely, if for any* PPT *adversary* $\mathcal{A}$ *the experiment* $\mathsf{Exp}^{\mathrm{tcca}}_{\mathcal{A}}(\lambda)$ *returns* 1 *with a probability bounded by* $\frac{1}{2} + \varepsilon_{TCCA}$*, then every* PPT *adversary wins the* q-TCCA *experiment with a probability bounded by* $\frac{1}{2} + q\varepsilon_{TCCA}$.

Proof. We build a sequence of $q+1$ hybrid games by modifying the rerandomization requests of the q-TCCA game in the following way. In the i-th hybrid, the game rerandomizes c_0 for the i-th first requests and c_1 for the remaining requests. It is clear that the first and last games correspond to the two games that the adversary tries to distinguish. We define S_i as the event that the adversary wins in the i-th hybrid.

We observe that an adversary $\mathcal{A}$ distinguishing the i-th hybrid from the $i+1$-th hybrid can be used to build a TCCA adversary $\mathcal{B}$ with identical advantage. $\mathcal{B}$ simulates $\mathcal{A}$ and answers its i-th first requests with a rerandomization of c_0. Then, it asks its oracle for the i-th request. Finally, it answers the remaining requests with randomized versions of c_1. It is clear that if $b = 0$, then $\mathcal{B}$ simulates the $i - th$ hybrid. Otherwise, it simulates the $i+1$-th hybrid. Hence $Pr[S_{i+1}] - Pr[S_i] \leq \varepsilon_{TCCA}$ for each $i \in [q-1]$, where ε_{TCCA} is the advantage of an efficient adversary playing the TCCA games. Finally,

$$|Pr[S_q] - Pr[S_0]| = |\sum_{i=0}^{q-1} Pr[S_{i+1} - S_i]| \leq q\epsilon,$$

which concludes the proof that TCCA implies q-TCCA. □

C Generic construction

We provide a generic construction of a secure extractable TREnc, denoted Π_G. It is mainly based on two primitives, which we introduce prior to specifying our TREnc: signatures on randomizable ciphertexts and tag-based encryption.

C.1 Signatures on Randomizable Ciphertexts

Signatures on Randomizable Ciphertexts (SRC), introduced by Blazy et al. [10], consists in a joint randomizable public-key encryption scheme and adaptable signature scheme on the ciphertexts and public key of the encryption scheme. These features can be used as an ingredient in order to obtain receipt-freeness, as they allow a third party to rerandomize a ciphertext and the associated signature, making it infeasible for the party who produced the original ciphertext to prove the content of the randomized ciphertext, while remaining able to trace the ciphertext thanks to the valid signature. However, by default, an SRC offers none of the privacy guarantees that we identified above.

Definition C.1 (Signatures on Randomizable ciphertexts). *A scheme of* signatures on randomizable ciphertexts *(SRC in short) consists of the following algorithms :* (Setup, $\mathsf{SKeyGen}$, $\mathsf{EKeyGen}$, Enc, Dec, Sign,

$\mathsf{Setup}(1^\lambda)$ *outputs the public parameter* pp *of the scheme, as defined in Sec. 4.1. Each of the following algorithms takes* pp *as an implicit argument.*

$(\mathsf{EKeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *represents an encryption scheme.* $\mathsf{EKeyGen}(1^\lambda)$ *outputs the pair* $(\mathsf{ek}, \mathsf{dk})$ *containing the public and secret keys of the encryption scheme.* $\mathsf{Enc}(\mathsf{ek}, \mathsf{vk}, m; r)$ *and* $\mathsf{Dec}(\mathsf{dk}, \mathsf{vk}, c)$ *respectively encrypts a message m with randomness r and decrypts a ciphertext c. The ciphertext can be signed with a signing key related to* vk.

$(\mathsf{SKeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *represents a signature scheme.* $\mathsf{SKeyGen}(1^\lambda)$ *outputs the pair* $(\mathsf{sk}, \mathsf{vk})$ *containing the signing and verification keys of the signature scheme.* $\mathsf{Sign}(\mathsf{sk}, \mathsf{ek}, c; s)$ *and* $\mathsf{Verify}(\mathsf{vk}, \mathsf{ek}, c, \sigma)$ *respectively signs a ciphertext and its encryption key, using randomness s, and verify the validity of a signature.*

$\mathsf{Rand}(\mathsf{ek}, \mathsf{vk}, c; r)$ *outputs a re-randomization of c using randomness r and* $\mathsf{Adapt}(\mathsf{vk}, \mathsf{ek}, c, \sigma; r, s)$ *outputs a new adapted signature.*

Besides the standard correctness properties of randomizable encryption schemes and signature schemes, we also require the scheme to follow the signature-adaptation *property: For every* ek, dk *in the range of* $\mathsf{EKeyGen}$*, every* vk, sk *in the range of* $\mathsf{SKeyGen}$*, message m and random coins r and s, for* $c = \mathsf{Enc}(\mathsf{ek}, \mathsf{vk}, m; r)$ *and* $\sigma = \mathsf{Sign}(\mathsf{sk}, \mathsf{ek}, c; s)$*,* $\{c' = \mathsf{Enc}(\mathsf{ek}, \mathsf{vk}, m; r'), \sigma' = \mathsf{Sign}(\mathsf{sk}, \mathsf{ek}, c'; s')\}$ *and* $\{c' = \mathsf{Rand}(\mathsf{ek}, \mathsf{vk}, c; r'), \sigma' = \mathsf{Adapt}(\mathsf{vk}, \mathsf{ek}, c, \sigma; r', s')\}$ *are statistically indistingui-shable.*

Definition C.2 (SRC Unforgeability). *An SRC is* unforgeable *if for every* PPT *adversary $\mathcal{A}$, the experiment* $\mathsf{Exp}^{\mathrm{uf}}_{\mathcal{A}}(\lambda)$ *defined in Figure 6 returns 1 with a probability negligible in λ.*

```
$\mathsf{Exp}^{\mathrm{src-uf}}_{\mathcal{A}}(\lambda)$
$(\mathsf{ek}, \mathsf{dk}) \leftarrow\$ \mathsf{EKeyGen}(1^\lambda)$
$(\mathsf{vk}, \mathsf{sk}) \leftarrow\$ \mathsf{SKeyGen}(1^\lambda)$
$(c, \sigma) \leftarrow\$ \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}, \mathsf{ek}, \cdot)}(\mathsf{vk}, (\mathsf{ek}, \mathsf{dk}))$
$m \leftarrow\$ \mathsf{Dec}(\mathsf{dk}, \mathsf{vk}, c)$
if $m \neq \perp$ and $m \notin Q$ and $\mathsf{Ver}(\mathsf{vk}, \mathsf{ek}, c, \sigma) = 1$
then return 1
```

Fig. 6. Unforgeability experiment. $\mathcal{A}$ has a signing oracle $\mathsf{Sign}(\mathsf{sk}, \mathsf{ek}, \cdot)$ that takes as input a ciphertext c. It outputs a signature on c with sk if this is a valid ciphertext and add $\mathsf{Dec}(\mathsf{dk}, c)$ to the list of used messages Q.

Both Blazy et al. [10] and Chaidos et al. [14] build their SRC on top of the Waters signature scheme, which is reminded in its variant on asymmetric pairings in Appendix E. A similar approach can be used to instantiate our generic construction, but we present a much more efficient solution in Section 4, that offers security in the standard model.

C.2 Tag-based encryption

Our generic TREnc construction starts from a tag-based encryption scheme, as proposed by Kiltz [27], where the tag will be the trace of the ciphertext.

Definition C.3. *A* tag-based encryption *scheme* $(\mathsf{TBGen}, \mathsf{TBEnc}, \mathsf{TBDec})$ *is a standard encryption scheme in which the encryption and decryption algorithms take an extra parameter t, that is,* $\mathsf{TBEnc}(\mathsf{pk}, t, m)$ *encrypts m with public key* pk *and tag t, and* $\mathsf{TBDec}(\mathsf{sk}, t, c)$ *decrypts c with secret key* sk *and tag t.*

We require our tag-based encryption scheme to offer weak CCA security in the sense of MacKenzie et al. [32].

Definition C.4 (TBE-wCCA). *A tag-based encryption scheme is said to be weakly secure against chosen ciphertext attacks (*TBE-wCCA*) if for every adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the experiment $\mathsf{Exp}_{\mathcal{A}}^{\text{tbe-wcca}}(\lambda)$ defined in Figure 7 returns* 1 *with a probability negligibly close in λ to $\frac{1}{2}$.*

```
$\mathsf{Exp}_{\mathcal{A}}^{\text{tbe-wcca}}(\lambda)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{TBGen}(1^\lambda)$
$(m_0, m_1, t^\star, \mathsf{st}) \leftarrow\$ \mathcal{A}_1^{\mathsf{TBDec}(\mathsf{sk},\cdot,\cdot)}(\mathsf{pk})$
$b \leftarrow\$ \{0,1\}$
$c^\star \leftarrow\$ \mathsf{TBEnc}(\mathsf{pk}, t^\star, m_b)$
$b' \leftarrow\$ \mathcal{A}_2^{\mathsf{TBDec}^\star(\mathsf{sk},\cdot,\cdot)}(c^\star, \mathsf{st})$
**return** $b = b'$
```

Fig. 7. Weakly CCA security experiment for tag-based encryption. In the experiment, $\mathcal{A}_2$ has access to a decryption oracle $\mathsf{TBDec}^\star$ which, on input (t, c), returns $\mathsf{TBec}(\mathsf{sk}, t, c)$ if $t \neq t^\star$ and $\mathtt{test}$ otherwise.

C.3 Building Blocks

All the following building blocks are assumed to satisfies a natural notion of verifiability in the sense of Definition 2.3.

A tag-based encryption scheme $(\mathsf{TBGen}, \mathsf{TBEnc}, \mathsf{TBDec})$.

We require the scheme to have two additional algorithms, a randomizing algorithm TBRand and a verification algorithm TBVer. More precisely, TBRand provides statistical randomizability for the same tag of a given ciphertext as long as it is in the range of honest encryptions. For verifiability: for any PPT adversary that receives a pair of public/secret keys as input and outputs a ciphertext c_{tbe}, the probability that $\mathsf{TBVer}(\mathsf{pk}_{tbe}, c_{tbe}) = 1$ and c_{tbe} is not in the range of $\mathsf{TBEnc}(\mathsf{pk}, \cdot, \cdot)$ is bounded by a negligible function ε_{TBVER}.

For the confidentiality, the tag-based encryption must be TBE-wCCA secure. More precisely, the advantage of any PPT adversary in the TBE-wCCA experiment (see Definition C.4) is bounded by a negligible function ε_{wCCA}.

A signature on ciphertexts scheme $(\mathsf{SGen}, \mathsf{Sign}, \mathsf{SVer})$.

Altogether with the tag-based encryption scheme, we require the signature on ciphertext scheme to form an SRC that has the signature adaptation property (regarding a signature adapting algorithm SAdapt). More precisely, the advantage of any unbounded adversary at distinguishing a randomization of a ciphertext-signature pair in the range of honestly generated pairs from a fresh re-encryption is negligible in λ. (See Definition C.1)

Additionally, we desire a signature scheme for which there exists an injective key derivation function ϕ that computes the verification key vk from the signing key sk: $\mathsf{vk} = \phi(\mathsf{sk})$. Since the key derivation function is injective, every verification key is associated to an unique signing key.

We further require the signature scheme to be verifiable in a stronger sense. More precisely, for any PPT adversary that takes as input the public parameters of SGen and outputs a signing key sk, an encryption key ek, a signature σ on a ciphertext c_{tbe}, the probability that $\mathsf{SVer}(\phi(\mathsf{sk}), \mathsf{ek}_{tbe}, c_{tbe}, \sigma) = 1$ and that σ is not in the range of $\mathsf{Sign}(\mathsf{sk}, \cdot, \cdot)$ is bounded by a negligible function ε_{SVER}.

Finally, the signature on ciphertexts scheme should be unforgeable: the advantage of any PPT adversary in the unforgeability experiment is bounded by a negligible function ε_{UF} (see Definition C.2).

A randomizable CPA-secure encryption scheme $(\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$.

The encryption scheme is required to be IND-CPA secure. More precisely, the advantage of any PPT adversary in the indistinguishability under chosen plaintext attacks experiment is bounded by a negligible function ε_{CPA}.

We also want the encryption scheme to have an additional randomizing algorithm Rand' and to be randomizable. That is, the advantage of any unbounded adversary at distinguishing a randomization of an honestly generated ciphertext from a fresh re-encryption is negligible in λ.

A NIZK malleable proof $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{PVer}, \mathsf{SimCRSGen}, \mathsf{SimProve})$.

The proof system proves statement of the form: $((\mathsf{ek}_{extr}, c_{extr}, \mathsf{vk}), (\mathsf{sk}, r)) \in R$ if and only if $\mathsf{vk} = \phi(\mathsf{sk})$ and $c_{extr} = \mathsf{Enc}(\mathsf{ek}_{extr}, \mathsf{sk}; r)$.

The proof system is required to be zero-knowledge. More precisely, the advantage of any PPT adversary at distinguishing between an honest CRS and a simulation CRS is bounded by a negligible function ε_{simcrs}. Likewise, the advantage of any PPT adversary at distinguishing between an honest prover and a simulated prover is bounded by a negligible function $\varepsilon_{simproof}$. We let $\varepsilon_{simcrs} + \varepsilon_{simproof} = \varepsilon_{ZK}$.

Moreover, the proof system should also be sound. More precisely, for any PPT adversary that receives an honestly generated crs as input and outputs a pair of statement/proof (x, π), the probability that x is not a true statement but $\mathsf{PVer}(\mathsf{crs}, x, \pi) = 1$ is bounded by a negligible function ε_{PSOUND}.

The proof system should also be verifiable. More precisely, for any PPT adversary that receives a crs as input and outputs a proof π for a statement x, the probability that $\mathsf{PVer}(\mathsf{crs}, x, \pi) = 1$ and π is not in the range of $\mathsf{Prove}(\mathsf{crs}, x, \cdot)$ is bounded by a negligible function ε_{PVER}.

Finally, we require the proofs generated by the proof system to be adaptable with regard to the randomizing procedure of the encryption (with an algorithm PAdapt). With regard to this malleability, the proof system should be derivation private. More precisely, the advantage of any unbounded adversary at distinguishing a derivation of a verified proof from a fresh proof of the same adapted statement is negligible in λ.

We stress that we do not need any form of simulation-soundness. In particular, we do not require the system to be simulation-sound extractable as defined for malleable proof systems in [15].

C.4 Description and security of the generic construction

Our generic extractable TREnc Π_G follows.

Setup(1^λ): Generate the public parameters pp of the building blocks.

Gen(1^λ, pp): Generate the public/secret keys $(\mathsf{ek}_{tbe}, \mathsf{dk}_{tbe}) \leftarrow \mathsf{TBGen}(1^\lambda, \mathsf{pp})$, $(\mathsf{ek}_{extr}, \mathsf{dk}_{extr}) \leftarrow \mathsf{Gen}'(1^\lambda, \mathsf{pp})$, and a CRS $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^\lambda, \mathsf{pp})$ of the proof system. Set $\mathsf{SK} = (\mathsf{dk}_{tbe})$, and $\mathsf{PK} = (\mathsf{ek}_{tbe}, \mathsf{ek}_{extr}, crs)$.

Enc(PK, m): This is computed as $\mathsf{LEnc}(\mathsf{PK}, \mathsf{LGen}(\mathsf{PK}), m)$.

LGen(PK): Generate a verifying/signing key $\mathsf{vk}, \mathsf{sk} \leftarrow\!\$\, \mathsf{SKeyGen}(1^\lambda, \mathsf{pp})$ of the SRC. Output $\mathsf{lk} = \mathsf{sk}$.

LEnc(PK, lk = sk, m):

1. Compute $\mathsf{vk} = \phi(\mathsf{sk})$, and $c_{tbe} \leftarrow \mathsf{TBEnc}(\mathsf{ek}_{tbe}, \mathsf{vk}, m)$.
2. Compute $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, c_{tbe})$ to get an SRC ciphertext.
3. Pick a random coin r and compute $c_{extr} = \mathsf{Enc}'(\mathsf{ek}_{\mathsf{extr}}, \mathsf{sk}; r)$.
4. Generate a proof $\pi \leftarrow \mathsf{Prove}(crs, (\mathsf{ek}_{extr}, c_{extr}, \mathsf{vk}), (\mathsf{sk}, r))$ given crs, the encrypted message sk inside c_{extr} under the randomness r, ensuring that $\mathsf{vk} = \phi(\mathsf{sk})$ and $c_{extr} = \mathsf{Enc}(\mathsf{ek}_{extr}, \mathsf{sk}; r)$.
5. Output the ciphertext $c = (\mathsf{vk}, c_{tbe}, \sigma, c_{extr}, \pi)$.

Ver(PK, CT): Run the verification algorithm of each scheme and output 1 if and only if they all returned 1.

Dec(sk, CT): If $\mathsf{Ver}(\mathsf{PK}, \mathsf{CT}) = 0$, output $\perp$. Otherwise, return $\mathsf{Dec}(\mathsf{dk}_{tbe}, \mathsf{vk}, c_{tbe})$.

Trace(PK, CT): If $\mathsf{Ver}(\mathsf{PK}, \mathsf{CT}) = 0$, output $\perp$. Otherwise, output vk.

Rand(PK, CT): Randomize c_{tbe}: $c'_{tbe} \leftarrow \mathsf{TBRand}(\mathsf{ek}_{tbe}, c_{tbe}; r'_{tbe})$ with random coin r'_{tbe} and adapt the signature: $\sigma' \leftarrow \mathsf{SAdapt}(\mathsf{vk}, \mathsf{ek}_{tbe}, \sigma, c_{tbe}; r'_{tbe}, s'_{tbe})$ with random coin s'_{tbe}. Randomize c_{extr} as well: $c'_{extr} \leftarrow \mathsf{Rand}'(\mathsf{ek}_{extr}, c_{extr}; r'_{extr})$. Adapt the proof: $\pi' \leftarrow \mathsf{PAdapt}(crs, \pi, r'_{extr})$. Return $(\mathsf{vk}, c'_{tbe}, \sigma', c'_{extr}, \pi')$.

Extractability:

TrapGen(1^λ, pp): Behave like $\mathsf{Gen}(1^\lambda, \mathsf{pp})$ but output $\mathsf{tk} = \mathsf{dk}_{extr}$ as well.

LExtr(lk, CT): Output $\mathsf{lk} = \mathsf{Dec}'(\mathsf{dk}_{extr}, c_{extr})$.

Theorem C.5. *Π_G is verifiable. More precisely, for any* PPT *adversary $\mathcal{A}$, we have $Pr[\mathsf{Exp}^{\mathrm{ver}}_{\mathcal{A},\Pi}(\lambda) = 1] \leq \varepsilon_{TBVER} + \varepsilon_{SVER} + \varepsilon_{PSOUND} + \varepsilon_{PVER}$.*

Proof. Let $\mathcal{A}$ be a PPT adversary. Given $\mathsf{PK}, \mathsf{SK} \leftarrow\!\$\, \mathsf{Gen}(1^\lambda)$, we have to show that the probability that $\mathcal{A}$ outputs a ciphertext CT such that $\mathsf{Ver}(\mathsf{PK}, \mathsf{CT}) = 1$ but that is not in the range of $\mathsf{Enc}(\mathsf{PK}, \cdot)$ is negligible. If $\mathsf{Ver}(\mathsf{PK}, \mathsf{CT}) = 1$ then $\mathsf{TBVer}(\mathsf{ek}_{tbe}, \mathsf{vk}, c_{tbe}) = 1$, $\mathsf{SVer}(\mathsf{vk}, \mathsf{ek}, c, \sigma) = 1$ and $\mathsf{PVer}(\mathsf{crs}, \pi, (\mathsf{ek}_{extr}, c_{extr}, \mathsf{vk})) = 1$. With a probability bounded by ε_{PSOUND}, π is a sound proof and $\mathsf{vk} = \phi(\mathsf{Dec}'(\mathsf{ek}_{extr}, c_{extr}))$. We note $\mathsf{sk} = \mathsf{Dec}'(\mathsf{dk}_{extr}, c_{extr})$. Also, with a probability bounded by $\varepsilon_{TBVER} + \varepsilon_{PVER}$, c_{tbe} is in the range of $\mathsf{TBEnc}(\mathsf{pk}, \mathsf{vk}, \cdot)$ and π is in the range of $\mathsf{Prove}(\mathsf{crs}, \cdot, \cdot)$. Finally, σ is in the range of $\mathsf{Sign}(\mathsf{sk}, \mathsf{ek}, c_{tbe})$ with a probability bounded by ε_{SVER}. Putting everything together, $Pr[\mathsf{Exp}^{\mathrm{ver}}_{\mathcal{A},\Pi}(\lambda) = 1] \leq \varepsilon_{TBVer} + \varepsilon_{SVER} + \varepsilon_{PSOUND} + \varepsilon_{PVER}$. □

Theorem C.6. *Π_G is traceable. More precisely, for any* PPT *adversary $\mathcal{A}$, we have $Pr[\mathsf{Exp}^{\mathrm{trace}}_{\mathcal{A},\Pi_G}(\lambda) = 1] \leq \varepsilon_{ZK} + \varepsilon_{CPA} + \varepsilon_{UF}$.*

Proof. The proof uses a sequence of games starting with the traceability experiment and ending with the unforgeability of signatures on ciphertexts experiment. We note S_i the event that the game outputs 1 in the i-th game. We show that S_0, the probability of an adversary to win the traceability experiment, is bounded by $\varepsilon_{ZK} + \varepsilon_{CPA} + \varepsilon_{UF}$.

$\mathsf{Game}_0(\lambda)$: Let $\mathcal{A}$ be a PPT adversary. We define $\mathsf{Game}_0(\lambda)$ as the original traceability experiment $\mathsf{Exp}^{\mathsf{trace}}_{\mathcal{A},\Pi_G}(\lambda)$ as defined in Definition 2.5. By definition, $\mathcal{A}$ wins the game with probability $Pr[S_0]$.

$\mathsf{Game}_1(\lambda)$: Instead of generating an honest CRS of the proof system, we use the $\mathsf{SimCRSGen}$ algorithm to set up a simulation CRS crs and a trapdoor τ that enables the generation of simulated proofs: $\mathsf{crs}, \tau \leftarrow\!\!\$\, \mathsf{SimCRSGen}(1^\lambda, \mathsf{pp})$

$\mathsf{Game}_0(\lambda) \rightarrow \mathsf{Game}_1(\lambda)$: From our assumption about the proof system, we have directly that $|Pr[S_0] - Pr[S_1]| \leq \varepsilon_{simcrs}$.

$\mathsf{Game}_2(\lambda)$: Instead of computing honestly the proof of the challenge ciphertext, we now use the simulation crs and the trapdoor to simulate the proof: $\pi \leftarrow\!\!\$\, \mathsf{SimProve}(\mathsf{crs}, \tau, (\mathsf{vk}, c_{extr}))$

$\mathsf{Game}_1(\lambda) \rightarrow \mathsf{Game}_2(\lambda)$: From our assumption about the proof system, we have directly that $|Pr[S_1] - Pr[S_2]| \leq \varepsilon_{simproof}$.

$\mathsf{Game}_3(\lambda)$: Instead of computing c_{extr} as an encryption of sk in the challenge ciphertext, we now encrypt a random signing key.

$\mathsf{Game}_2(\lambda) \rightarrow \mathsf{Game}_2(\lambda)$: Instead of computing ourselves the part c_{extr} of the challenge, we could instantiate an IND-CPA experiment and use the challenger to encrypt either sk (in $\mathsf{Game}_1(\lambda)$) or a random signing key (in $\mathsf{Game}_2(\lambda)$). The proof that the ciphertext is well constructed is simulated from the previous game. Hence any distinguisher between $\mathsf{Game}_1(\lambda)$ and $\mathsf{Game}_2(\lambda)$ could be used to create an IND-CPA adversary and $|Pr[S_2] - Pr[S_3]| \leq \varepsilon_{CPA}$.

$\mathsf{Game}_4(\lambda)$: We transform $\mathsf{Game}_3(\lambda)$ into $\mathsf{Game}_4(\lambda)$ by modifying the way the ciphertext is created. In this game, we do not generate the pair of public/secret keys anymore and instantiate an unforgeability of signatures on ciphertexts experiment instead. We receive from the unforgeability challenger a verification key vk and a pair of public/secret key $(\mathsf{ek}_{tbe}, \mathsf{dk}_{tbe})$ of the tag-based encryption scheme. Then we generate the remaining parts of the TREnc keys with the Gen' and the trapdoor CRS generation algorithm, send the public and secret keys to $\mathcal{A}$ and receive a message m from $\mathcal{A}$. We encrypt m with TBGen and send the resulting ciphertext as a signing request to the unforgeability challenger. We compute c_{extr} and π as in the previous game.

$\mathsf{Game}_2(\lambda) \rightarrow \mathsf{Game}_3(\lambda)$: Everything is computed as before but through the challenger. $Pr[S_3] = Pr[S_4]$.

Finally, $\mathsf{Game}_3(\lambda)$ is equivalent to the unforgeability experiment. Since the scheme is unforgeable, $Pr[4]$ is bounded by $\mathsf{negl}_{UF}(\lambda)$. Thus $Pr[S_0] \leq \varepsilon_{ZK} + \varepsilon_{CPA} + \varepsilon_{UF}$ and the scheme is traceable. $\square$

Theorem C.7. *$\mathit{\Pi}_G$ is strongly randomizable. More precisely, for every $c \in \mathsf{LEnc}(\mathsf{pk}, \mathsf{lk}, m)$ with pk in the range of Gen and lk in the range of $\mathsf{LGen}(\mathsf{pk})$, the following computational indistinguishability relation holds:*

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

Proof. We parse $\mathsf{pk} = (\mathsf{ek}_{tbe}, \mathsf{ek}_{extr}, crs)$ and c as $(\mathsf{vk}, c_{tbe}, \sigma, c_{extr}, \pi)$ where $\mathsf{vk} = \phi(\mathsf{lk})$ (otherwise c would not be in the range of $\mathsf{LEnc}(\mathsf{pk}, \mathsf{lk}, \cdot)$).

Since (c_{tbe}, σ) is in the range of $(\mathsf{TBEnc}(\mathsf{ek}_{tbe}, \mathsf{vk}, m), \mathsf{Sign}(\mathsf{lk}, c_{tbe}))$, and the tag-based encryption scheme coupled to the signature scheme form a SRC, the signature adaptation property guarantees that $\{\mathsf{TBEnc}(\mathsf{ek}_{tbe}\phi(\mathsf{lk}), m; r'), \mathsf{Sign}(\mathsf{lk}, \mathsf{ek}_{tbe}, c'; s')\}$ and $\{\mathsf{TBRand}(\mathsf{ek}_{tbe}, \mathsf{vk}, c; r'), \mathsf{SAdapt}(\mathsf{vk}, \mathsf{ek}_{tbe}, c_{tbe}, \sigma; r', s')\}$ are indistinguishable.

Likewise, $\{\mathsf{Enc}'(\mathsf{ek}_{extr}, \mathsf{lk}; r'_{extr}), \mathsf{Prove}(crs, (\mathsf{ek}_{extr}, c_{extr}, \mathsf{vk}), (\mathsf{lk}, r'_{extr}))\}$ is computationally indistinguishable from $\{\mathsf{Rand}'(\mathsf{ek}_{extr}, c_{extr}; r'_{extr}), \mathsf{PAdapt}(crs, \pi, r'_{extr})\}$ because the encryption scheme is randomizable and the proof system is adaptable. $\square$

Theorem C.8. *$\mathit{\Pi}_G$ is extractable. More precisely, $\mathit{\Pi}_G$ satisfies the trapdoor indistinguishability and unique extractability properties.*

Proof. **Trapdoor indistinguishability** The public and secret keys are computed in the exact same way in Gen and $\mathsf{TrapGen}$.
Unique Extraction Since $(\mathsf{pk}, \mathsf{sk}, \mathsf{tk})$ is in the range of $\mathsf{TrapGen}$, the second entry of pk and tk form a pair of public and secret keys $(\mathsf{pk}_{extr}, \mathsf{dk}_{extr})$ of the CPA-secure scheme.

- If a ciphertext c is in the range of $\mathsf{LEnc}(\mathsf{pk}, \mathsf{lk}, \cdot)$, then the c_{extr} part of c is also in the range of $\mathsf{Enc}'(\mathsf{ek}_{extr}, \mathsf{lk})$. Because Enc' and Dec' are part of a CPA-secure encryption scheme, $\mathsf{LExtr}(c, \mathsf{tk}) = \mathsf{Dec}'(\mathsf{dk}_{extr}, c_{extr}) = \mathsf{lk}$.
- Since c_0 and c_1 are both in the range of $\mathsf{Enc}(\mathsf{pk}, \cdot)$, there are two link keys lk_0 and lk_1 in the range of $\mathsf{LGen}(\mathsf{pk})$ such that $c_i \in \mathsf{LEnc}(\mathsf{pk}, \mathsf{lk}_i, \cdot)$ for $i = 0, 1$. Also, they both have a component vk_i such that $\mathsf{vk}_i = \phi(\mathsf{lk}_i)$. Since $\mathsf{Trace}(\mathsf{pk}, c_0) = \mathsf{Trace}(\mathsf{pk}, c_1)$, $\mathsf{vk}_0 = \mathsf{vk}_1$ and the injectivity of ϕ garantees that $\mathsf{lk}_0 = \mathsf{lk}_1$. Finally, the first item of the definition implies that $\mathsf{LExtr}(\mathsf{tk}, c_0) = \mathsf{lk}_0 = \mathsf{lk}_1 = \mathsf{LExtr}(\mathsf{tk}, c_1)$.

$\square$

Theorem C.9. *$\mathit{\Pi}_G$ is* pd*-RCCA*. More precisely, for any adversary* PPT *$\mathcal{A}$, we have $Pr[\mathsf{Exp}^{\text{pd}^\star\text{-rcca}}_{\mathcal{A},\mathit{\Pi}_G}(\lambda) = 1] \leq \varepsilon_{wCCA}$.*

Proof. The proof is a direct reduction. Let $\mathcal{A}$ be an adversary for the pd*-RCCA experiment winning with an advantage greater than $\varepsilon_{wCCA}(\lambda)$. Using $\mathcal{A}$, we build an adversary $\mathcal{B}$ for the TBE-wCCA experiment that also wins with the same advantage.

At the beginning of the experiment, $\mathcal{B}$ receives from the challenger the pair of public/private key of the tag-based encryption scheme used by the challenger. In addition to these keys, $\mathcal{B}$ computes

$\mathsf{ek}_{extr}, \mathsf{dk}_{extr}$ and crs to generate the public, secret and trapdoor keys of the TREnc used to interact with a pd*-RCCA adversary. $\mathcal{B}$ sends the public key to $\mathcal{A}$. For each of the decryption request that it receives from $\mathcal{A}$, $\mathcal{B}$ verifies that the ciphertext is valid. It then sends vk, c_{tbe} to the TBE-wCCA challenger and returns the output to $\mathcal{A}$.

At some point, $\mathcal{B}$ receives the challenge (m_0, m_1, lk) from $\mathcal{A}$. If $\mathsf{lk} = \perp$, it generates itself a link key by running the $\mathsf{LEnc}(\mathsf{PK})$ algorithm and computes the verification key $\mathsf{vk} = \phi(\mathsf{lk})$. $\mathcal{B}$ chooses (m_0, m_1, vk) as its challenge and receives a tag-based encryption $c^\star$ of either m_0 or m_1 from the challenger. It generates a signature σ on $c^\star$ using sk and computes an encryption of sk as well as the accompanying proof respectively with $c_{extr} = \mathsf{Enc}'(\mathsf{ek}_{extr}, \mathsf{sk}; r)$ and $\pi \leftarrow\!\$\ \mathsf{Prove}(\mathsf{crs}, (\mathsf{ek}_{extr}, c_{extr}, \mathsf{vk}), (\mathsf{sk}, r))$. $\mathsf{CT}^\star = (\mathsf{vk}, c^\star, \sigma, c_{extr}, \pi)$ forms a challenge ciphertext that has the same distribution as ciphertexts directly computed with the Enc algorithm of the TREnc.

$\mathcal{B}$ deals with the post-challenge decryption requests exactly as before, except if the ciphertext has the same trace as $\mathsf{CT}^\star$. In that case, it returns test to $\mathcal{A}$. Especially, it never asks the challenger for a decryption using the challenge trace vk since such a ciphertext would have the same trace as $\mathsf{CT}^\star$. Finally, when $\mathcal{A}$ sends a guess b', $\mathcal{B}$ returns it to the challenger.

As the probability of $\mathcal{B}$ to win the TBE-wCCA experiment is the same as the probability of $\mathcal{A}$ to win the pd*-RCCA experiment, we have a contradiction and the advantage of $\mathcal{A}$ has to be bounded by $\varepsilon_{wCCA}(\lambda)$. $\square$

Since Π_G is extractable and pd*-RCCA-secure, it follows from Theorem B.1 that Π_G is TCCA-secure as well.

C.5 Instantiation

To build an extractable TREnc, we briefly sketch how to instantiate each building blocks in asymmetric bilinear groups. For the SRC, we start from BeleniosRF and use the Waters' signature scheme adapted to asymmetric pairings [10], see Appendix E.[4] For the weak CCA secure randomizable tag-based encryption, we depart from BeleniosRF [14] as their TBE is only selective-tag weakly CCA secure. While it is easy to show that this enough to generically implies RCCA security, it is not enough to reach our natural variant of pd-RCCA, even without extractor, which is necessary to get TCCA security. We thus borrow the scheme due to [32], which is the Cramer-Shoup ciphertext where the hashed value is replaced by the tag. It is easy to see that this provides our desired notion of randomizability and that we can turn it into a randomizable SRC by adding the same additional components of [14] to allow adapting the signature after the randomization of the TBE. However, we also have to turn it into a publicly verifiable TBE/CRS, and include a randomizable simulation-sound proof that we indeed encrypt the Waters' hash of the signed message that is decomposed bit by bit. As in BeleniosRF, we can rely on the Groth-Sahai proof system which provides perfect NIWI proofs for quadratic statement of pairing-product equations (when the CRS is generated in the hiding mode). To turn them into a simulation-sound ZK proof, we can combine them

[4] It is easy to see that the signature has all the security guarantees that we want. Valid signatures are always in the range of honest signatures, and the public key unequivocally defines the secret key.

with a structure-preserving signature on Waters' public key, and prove that either the statement is true or that we have such a valid signature as suggested in [15].[5]

The remaining part is more straightforward. We can take a perfectly randomiz-able ElGamal encryption to encrypt the Waters' signing key (which is a single group element) and use its decryption key for extraction. For the randomizable NIZK proof that we indeed faithfully encrypt the signing key, we can take the same modified Groth-Sahai proof system described above.

D Proofs of security of the construction under SXDH

D.1 Verifiability

Theorem D.1. *The above TREnc is verifiable under the SXDH assumption. More precisely, for any adversary $\mathcal{A}$, we have* $\Pr[\mathsf{Exp}^{\mathsf{ver}}_{\mathcal{A}}(\lambda) = 1] \leq \varepsilon_{\mathsf{sxdh}} + 1/p$.

Proof. Let $\mathcal{A}$ be an efficient adversary. Given $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{Gen}(1^\lambda)$, we have to show that any ciphertext $\mathsf{CT} \leftarrow \mathcal{A}(\mathsf{PK}, \mathsf{SK})$ which passes the verification equations so that $\mathsf{Ver}(\mathsf{PK}, \mathsf{CT}) = 1$ is necessarily in the range of the honestly generated encryptions of $\mathsf{Dec}(\mathsf{SK}, \mathsf{CT})$ that traces to $\mathsf{Trace}(\mathsf{PK}, \mathsf{CT})$.

During the key generation we keep δ such that $F = f^\delta$, and $G = g^\delta$. We also keep the secret keys sk_u and sk_v to compute one-time LHSP signatures. We denote by $\mathsf{sk}^{\mathsf{qazk}}_{\mathsf{lhsp}}$ the concatenation of these keys which corresponds to the signing key associated to the public key $\mathsf{pk}^{\mathsf{qazk}}_{\mathsf{lhsp}} = \{\hat{u}_1, \hat{u}_2, \hat{v}_1, \hat{v}_2\}$. We recall that only $\mathsf{SK} = (\alpha, \beta)$ that satisfies $f = g^\alpha h^\beta$ is the secret key. The other public elements compose the CRS of the scheme.

Now, let $(\mathbf{c}, \boldsymbol{C}_Z, \boldsymbol{C}_R, \sigma_2, \sigma_3, \pi, \hat{\pi}_{\mathsf{sig}}, \mathsf{opk}) = \mathsf{CT} \leftarrow \mathcal{A}(\mathsf{PK}, \mathsf{SK})$ that satisfies $\mathsf{Ver}(\mathsf{PK}, \mathsf{CT}) = 1$, where we have $\mathbf{c} = (c_0, c_1, c_2)$, $\pi = (Z_\pi, R_\pi)$, $\hat{\pi}_{\mathsf{sig}} = (\hat{P}_1, \hat{P}_2)$ and $\mathsf{Trace}(\mathsf{PK}, \mathsf{CT}) = \mathsf{opk} = (\hat{f}_1, \hat{f}_2, \hat{f}_3)$.

First, we show that the validity of $\pi = (Z_\pi, R_\pi)$ ensures the existence of $m \in \mathbb{G}$ and $\theta \in \mathbb{Z}_p$ such that $c_0 = m \cdot f^\theta$, $c_1 = g^\theta$ and $c_2 = h^\theta$. Indeed, the verification equation (4) implies that π is a valid one-time LHSP signature on the vector $(c_1^\tau, c_2^\tau, c_1, c_2)$, where $\tau = H(\mathsf{opk})$. Now, we assume that (c_1, c_2) is not in the span generated by (g, h). That is $c_1 = g^{\theta_1}$ and $c_2 = h^{\theta_2}$, for some $\theta_1, \theta_2 \in \mathbb{Z}_p$ with $\theta_1 \neq \theta_2$. Therefore, the vector $(c_1^\tau, c_2^\tau, c_1, c_2) = (g^{\theta_1 \tau}, h^{\theta_2 \tau}, g^{\theta_1}, h^{\theta_2})$ is not in the span generated by the vectors $(g, h, 1, 1)$ and $(1, 1, g, h)$ that are signed in PK under $\mathsf{pk}^{\mathsf{qazk}}_{\mathsf{lhsp}}$. By the property of the LHSP signature scheme, the honest deterministic signature on $(c_1^\tau, c_2^\tau, c_1, c_2)$ that would use $\mathsf{sk}^{\mathsf{qazk}}_{\mathsf{lhsp}}$ remains statistically unpredictable. That is because $\mathsf{sk}^{\mathsf{qazk}}_{\mathsf{lhsp}}$ still contains enough entropy that are only revealed when signing vectors outside the span of the vectors that are already signed. As a consequence, $(Z^\dagger, R^\dagger) = \mathsf{Sign}(\mathsf{sk}^{\mathsf{qazk}}_{\mathsf{lhsp}}, (c_1^\tau, c_2^\tau, c_1, c_2))$ also satisfies the verification equation (4) with the same right-hand side member, and $Z^\dagger \neq Z_\pi$ and $R^\dagger \neq R_\pi$, but with probability 1/p. By dividing both equations we get $e(Z^\dagger / Z_\pi, \hat{g}) \cdot e(R^\dagger / R_\pi, \hat{h}) = 1$, which implies a DP solver and thus an SXDH distinguisher. To conclude this step, there exists $\theta \in \mathbb{Z}_p$ such that $c_1 = g^\theta$ and $c_2 = h^\theta$, except with negligible probability. Moreover, we can write $c_0 = m \cdot f^\theta$, for some $m \in \mathbb{G}$, so that $\mathsf{Dec}(\mathsf{SK}, \mathsf{CT}) = m$. Eventually, even if we do not know θ, π must be equal to $\pi^\dagger := (Z^\dagger, R^\dagger) = (\Sigma_u^\tau \Sigma_v)^\theta$ as otherwise the same reduction to solving the DP instance $(\hat{g}, \hat{h})$ would work. Indeed, it is enough to have distinct

[5] In BeleniosRF, there is no such a composition and the proofs seems to only be WI.

valid signatures on the same vector to solve the problem. By the way, we also showed that π has the right distribution.

Second, we show that the Groth-Sahai commitments $\boldsymbol{C}_Z$ and $\boldsymbol{C}_R$ and the proof $\hat{\pi}_{\mathsf{sig}}$ are reachable by an honest run of the encryption algorithm for some random coin. Since crs_w is generated in the perfect NIWI mode, the resulting commitments and proofs are distributed among all the possible group elements that satisfy the verification equation. That means that, as long as the adversarially generated $\boldsymbol{C}_Z$, $\boldsymbol{C}_R$ and $\hat{\pi}_{\mathsf{sig}}$ are valid, they can necessarily be explained by an honest computation given the group elements involved in the verification equation $(\hat{g}, \hat{h}, g, c_0, c_1)$ and opk. We stress that we can explained the committed signature $\sigma_1 = (Z_1, R_1)$ by a random pair of group elements since the commitments are perfectly hiding and the CRS crs_w forms a basis of $\mathbb{G}^2$. That is, the view is actually independent of any honest σ_1, which therefore does not give any constraint on the LHSP signatures for the tracing part. Moreover, an honest opk is uniform in $\mathbb{G}^3$, so any triple is reachable for this tracing value.

Third, we show that opk and the associated LHSP signatures σ_2 and σ_3 related to the tracing part of the ciphertext can also be explained as an honest run of the encryption algorithm. We are looking for some $\mathsf{lk} = \mathsf{osk} = \{(\eta_i, \zeta_i)\}_{i=1}^3$ such that $\hat{f}_i = \hat{g}^{\eta_i}\hat{h}^{\zeta_i}$, for $i = 1$ to 3, and that also satisfy $\sigma_j = \mathsf{Sign}(\mathsf{osk}, \vec{T_j})$, for $j = 1$ to 2, where $\vec{T_j} = (T_{j,1}, T_{j,2}, T_{j,3})$ is the j-th row of the matrix $\mathbf{T}$ of equation (2). Since $\vec{T_3} = (1, F, G) = (1, f, g)^\delta = \vec{T}_2^\delta$, we must have $\sigma_3 = \sigma_2^\delta$ if they were honestly computed. However, if $\sigma_3 \neq \sigma_2^\delta$ we would have two distinct LHSP signatures on the vector $\vec{T_3}$, which again implies a DP solver and thus an SXDH distinguisher. We note that any group elements can be reached by the public key and the signatures (up to the validity of the verification equation) during an honest execution of the LHSP algorithms as they are all uniform constrained to the validity. We thus only have one signature σ_2 that imposes a linear constraint on the secret key osk (since the Groth-Sahai proof does not reveal any purported honest σ_1) while opk leaves 3 degrees of freedom. It remains 2 degrees of freedom to define all the possible honest secret keys osk corresponding to the view. This is largely enough to assert that this tracing part is in the range of an honest encryption.

In conclusion, a valid ciphertext CT is in the range of all the honestly generated encryption of $\mathsf{Dec}(\mathsf{SK}, \mathsf{CT})$ that traces to $\mathsf{Trace}(\mathsf{PK}, \mathsf{CT})$, except if the proof $\pi \neq \mathsf{Sign}(\mathsf{sk}_{\mathsf{lhsp}}^{\mathsf{qazk}}, (c_1^\tau, c_2^\tau, c_1, c_2))$ or the signature $\sigma_3 \neq \sigma_2^\delta$ that are both related to the public parameters $(\hat{g}, \hat{h})$. Given a single DP problem instance $(\hat{g}, \hat{h})$ it is straightforward to build an efficient adversary $\mathcal{B}$ that solves it with the same probability as $\mathcal{A}$ produces a valid CT with at least one of these inequalities. We thus have $\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathrm{ver}}(\lambda) = 1] \leq \varepsilon_{\mathsf{sxdh}} + 1/p$. □

D.2 Traceability

Theorem D.2. *The above TREnc Π is traceable under the SXDH assumption. More precisely, for any adversary $\mathcal{A}$, we have* $\Pr[\mathsf{Exp}_{\mathcal{A},\Pi}^{\mathrm{trace}}(\lambda) = 1] \leq 2 \cdot \varepsilon_{\mathsf{sxdh}} + 2/p$.

Proof. Let $\mathcal{A}$ be an efficient adversary against the traceability of our TREnc Π. We consider a sequence of games. In Game i, we denote by S_i the event that $\mathcal{A}(\mathsf{PK}, \mathsf{SK})$ wins by producing a valid ciphertext $\mathsf{CT}^* = (\mathbf{c}^*, \boldsymbol{C}_Z^*, \boldsymbol{C}_R^*, \sigma_2^*, \sigma_3^*, \pi^*, \hat{\pi}_{\mathsf{sig}}^*, \mathsf{opk}^*)$ that traces to $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{PK}, m)$ but $\mathsf{Dec}(\mathsf{SK}, \mathsf{CT}^*) \neq m$, where the message m was chosen by the adversary.

Game 0: This is the real game, where $(\mathsf{PK},\mathsf{SK}) \leftarrow \mathsf{Gen}(1^\lambda)$ with $\mathsf{SK} = (\alpha,\beta)$ and $\mathsf{PK} = (f, g, h, F, G, \mathsf{crs}_w, \Sigma_u, \Sigma_v, \mathsf{pk}$
Then, $(m, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{PK},\mathsf{SK})$, $(\mathbf{c}, \boldsymbol{C}_Z, \boldsymbol{C}_R, \sigma_2, \sigma_3, \pi, \hat{\pi}_{\mathsf{sig}}, \mathsf{opk}) := \mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{PK}, m)$, and $\mathsf{CT}^* \leftarrow \mathcal{A}_2(\mathsf{st}, \mathsf{CT})$. By definition, the event S_0 occurs if $\mathsf{Ver}(\mathsf{PK}, \mathsf{CT}^*) = 1$, $\mathsf{Dec}(\mathsf{SK}, \mathsf{CT}^*) \neq m$, and $\mathsf{opk}^* = \mathsf{Trace}(\mathsf{PK}, \mathsf{CT}^*) = \mathsf{Trace}(\mathsf{PK}, \mathsf{CT}) = \mathsf{opk}$. We thus have $\Pr[S_0] = \Pr[\mathsf{Exp}^{\mathsf{trace}}_{\mathcal{A},\Pi}(\lambda) = 1]$.

Game 1: This game is as the real game except that the Groth Sahai CRS $\mathsf{crs}_w = (\vec{w}_1, \vec{w}_2)$ of the public key is now generated in the extractable mode. Namely, instead of picking $\vec{w}_1, \vec{w}_2 \leftarrow\!\!\$\, \mathbb{G}^2$ uniformly at random, we pick $\vec{w}_1 \leftarrow\!\!\$\, \mathbb{G}^2$ and $\xi \leftarrow\!\!\$\, \mathbb{Z}_p$, and compute $\vec{w}_2 = \vec{w}_1^{\xi}$. Now, the CRS crs_w forms a random DH tuple over $\mathbb{G}$.

Any distinct behavior of $\mathcal{A}$ between Game 0 and Game 1 leads to an SXDH distinguisher (DDH in $\mathbb{G}$). We have, $|\Pr[S_0] - \Pr[S_1]| \leq \varepsilon_{\mathsf{sxdh}}$.

Game 2: We bring the following modification to the previous game. When we sample $\mathsf{crs}_w = (\vec{w}_1, \vec{w}_2)$ as a random DH tuple over $\mathbb{G}$, we first pick $w_{11} \leftarrow\!\!\$\, \mathbb{G}$ and $\gamma \leftarrow\!\!\$\, \mathbb{Z}_p$ to compute $\vec{w}_1 = (w_{11}, w_{12})$, where $w_{12} = w_{11}^{\gamma}$. The distribution of the public key is unchanged, but we keep γ as an ElGamal secret key to extract the committed group elements of the Groth-Sahai commitments. More precisely, given $\boldsymbol{C}_Z^*, \boldsymbol{C}_R^* \in \mathbb{G}^2$, we extract/decrypt some $Z_1^*, R_1^* \in \mathbb{G}$. Now, we abort and output 0 if $\sigma_1^* := (Z_1^*, R_1^*)$ is not a valid one-time LHSP signature on (g, c_0^*, c_1^*), where $\boldsymbol{c}^* = (c_0^*, c_1^*, c_2^*)$ from CT^*, under $\mathsf{opk}^* = \mathsf{opk}$.

The perfect soundness of the Groth-Sahai proofs in the extractable mode ensures that the probability to abort in the case of a invalid $\sigma_1^* := (Z_1^*, R_1^*)$ is 0. We thus find that $\Pr[S_1] = \Pr[S_2]$.

Game 3: This game is as Game 2 except in the way we consider $\mathcal{A}$ successful. First, when we generate PK, we keep $\mathsf{sk}^{\mathsf{qazk}}_{\mathsf{lhsp}} := (\mathsf{sk}_u, \mathsf{sk}_v)$ associated to $\mathsf{pk}^{\mathsf{qazk}}_{\mathsf{lhsp}} = \{\hat{u}_1, \hat{u}_2, \hat{v}_1, \hat{v}_2\}$ and that was used to compute Σ_u and Σ_v. Second, when we create the encryption CT of the message m given by $\mathcal{A}_1(\mathsf{PK},\mathsf{SK})$, we first pick $\mathsf{lk} \leftarrow \mathsf{LGen}(\mathsf{PK})$ ourself. That is, we honestly generate a private key of the LHSP signature scheme $\mathsf{lk} = \mathsf{osk} = \{(\eta_i, \zeta_i)\}_{i=1}^3$ and compute the corresponding public key $\mathsf{opk} = \{\hat{f}_i\}_{i=1}^3$. Then, we honestly generate $\mathsf{LEnc}(\mathsf{PK}, \mathsf{osk}, m)$ to get $\mathsf{CT} = (\mathbf{c}, \boldsymbol{C}_Z, \boldsymbol{C}_R, \sigma_2, \sigma_3, \pi, \hat{\pi}_{\mathsf{sig}}, \mathsf{opk})$, but we keep osk. As soon as we get CT^* from adv_2, we extract $\sigma_1^* := (Z_1^*, R_1^*)$ as in Game 2. Now, we also abort and output 0 if $\sigma_1^* \neq \mathsf{Sign}(\mathsf{osk}, (g, c_0^*, c_1^*))$ or $\pi^* \neq \mathsf{Sign}(\mathsf{sk}^{\mathsf{qazk}}_{\mathsf{lhsp}}, (c_1^{*\tau}, c_2^{*\tau}, c_1^*, c_2^*))$, where $\tau = H(\mathsf{opk})$.

The view of Game 2 and Game 3 have the same distribution as long as we do not consider $\mathcal{A}$ unsuccessful, and somehow reject the validity of CT^*, in Game 3 while $\mathcal{A}$ would have been deemed successful in Game 2. Therefore, $|\Pr[S_2] - \Pr[S_3]|$ is bounded by the probability that $\mathcal{A}$ (implicitly) produces a valid $\sigma_1^* \neq \mathsf{Sign}(\mathsf{osk}, (g, c_0^*, c_1^*))$ or a valid $\pi^* \neq \mathsf{Sign}(\mathsf{sk}^{\mathsf{qazk}}_{\mathsf{lhsp}}, (c_1^{*\tau}, c_2^{*\tau}, c_1^*, c_2^*))$. However, given the public parameter $(\hat{g}, \hat{h})$ of the LHSP scheme, and given any 2 distinct one-time LHSP signatures on a same vector, we have straightforward reduction to a DP solver. Hence, $|\Pr[S_2] - \Pr[S_3]| \leq \varepsilon_{\mathsf{sxdh}}$.

Now we conclude by showing that $\Pr[S_3] = 2/p$. We use the same argument about the one-time LHSP security scheme (recalled in Section 4.2) that we used in the proof of verifiability, Theorem D.1. First, we show that in Game 3 (c_1^*, c_2^*) is of the form $(g, h)^{\theta^*}$ except with probability 1/p. That is, $\boldsymbol{c}^* = (m^* f^{\theta^*}, g^{\theta^*}, h^{\theta^*})$, for some $m^* \in \mathbb{G}$, and then $m^* = \mathsf{Dec}(\mathsf{SK}, \mathsf{CT}^*)$. Indeed, PK only contains one-time LHSP signatures Σ_u and Σ_v on the vectors $(g, h, 1, 1)$ and $(1, 1, g, h)$, respectively. Then, if $\boldsymbol{c}^*$ is not an honest CPE encryption of m^*, $(c_1^{*\tau}, c_2^{*\tau}, c_1^*, c_2^*)$ is not in the span generated by the vectors $(g, h, 1, 1)$ and $(1, 1, g, h)$, and the honest one-time LHSP signature $\pi^\dagger := \mathsf{Sign}(\mathsf{sk}_{\mathsf{lhsp}}^{\mathsf{qazk}}, (c_1^{*\tau}, c_2^{*\tau}, c_1^*, c_2^*))$ remains unpredictable even in front of an unbounded adversary. Then, the probability to guess the right $\pi^\dagger = (Z^\dagger, R^\dagger)$ so that $\pi^* := (Z^*, R^*) = (Z^\dagger, R^\dagger)$ is equal to the probability to have $Z^* = Z^*$ since the verification equation would necessarily implies $R^* = R^*$ in that case. But, the probability to have $Z^* = Z^*$ is $1/p$. Finally, since $\boldsymbol{c}$ and $\boldsymbol{c}^*$ are CPA encryptions of distinct messages m and m^* respectively, the vector $(g, c_0^*, c_1^*) = (g, m^* f^{\theta^*}, g^{\theta^*})$ is necessarily outside the span generated by the vectors $(g, c_0, c_1) = (g, mf^\theta, g^\theta)$, $(1, f, g)$ and $(1, F, G)$ (for which we provide the one-time LHSP signatures σ_1, σ_2 and σ_3) which is of dimension 2 since $(1, F, G) = (1, f, g)^\delta$ (and $\sigma_3 = \sigma_2^\delta$). Indeed, (g, c_0^*, c_1^*), (g, c_0, c_1) and $(1, f, g)$ form a basis of $\mathbb{G}^3$ as long as $m^* \neq m$. Therefore, by a similar argument as above the probability to guess the honest one-time LHSP signature $\sigma_1^\dagger := \mathsf{Sign}(\mathsf{osk}, (g, c_0^*, c_1^*))$ is also $1/p$ (since σ_3 contains no more information than σ_2 about osk).

Eventually, $\Pr[\mathsf{Exp}_{\mathcal{A},\Pi}^{\mathrm{trace}}(\lambda) = 1] = \Pr[S_0] \leq 2 \cdot \varepsilon_{\mathsf{sxdh}} + 2/p$, as expected. □

D.3 Strong Randomizability

Theorem D.3. *The above TREnc is* perfectly *strongly randomizable.*

Proof. Let CT be a ciphertext in the range of $\mathsf{LEnc}(\mathsf{PK}, \mathsf{osk}, m)$, where the link key is $\mathsf{osk} = \{(\eta_i, \zeta_i)\}_{i=1}^3$ and the associated trace is $\mathsf{opk} = \{\hat{f}_i\}_{i=1}^3$. Even if the coin explaining c might have been taken from any dishonest distribution, we show that the randomization of CT is indistinguishable from a fresh and honest linked encryption of m with osk.

In item 2 of the encryption algorithm, the CPA encryption $\mathbf{c} = (c_0, c_1, c_2) = (mf^\theta, g^\theta, h^\theta)$ is perfectly randomizable for any original coin $\theta \in \mathbb{Z}_p$ that explains $\mathbf{c}$ since $\mathbf{c}' = (c_0, c_1, c_2) \cdot (f, g, h)^{\theta'}$ is equal to $(mf^{\theta+\theta'}, g^{\theta+\theta'}, h^{\theta+\theta'})$ which is distributed exactly as a fresh CPA encryption of m since $\theta + \theta'$ is random over $\mathbb{Z}_p$. In the same manner, it is easy to see that $\pi = (\Sigma_u^\tau \Sigma_v)^\theta$ is also perfectly strongly randomizable, where $\tau = H(\mathsf{opk})$ is a constant in both distributions.

By definition, Rand does not modify σ_2, σ_3. Moreover, the signing algorithm is deterministic, so that given osk that explains σ_2, σ_3 as signatures on the last two rows of $\mathbf{T}$ of equation (2) that are fixed by PK, any fresh run of the signing algorithm with osk gives that them back. As σ_1 only appears implicitly in the Groth-Sahai commitments $\boldsymbol{C}_Z, \boldsymbol{C}_R$ and proof $\hat{\pi}_{\mathsf{sig}}$ in the perfect NIWI setting, the perfect randomization holds even after the adaptation in item 2 of Rand. That is re-randomized (linear) Groth-Sahai (commitments and) proofs are perfectly re-distributed among all the (commitments and) valid proofs that satisfy verification equations that are, here, chosen with the same distribution parametrized by the c_0 and c_1 components. □

D.4 Traceable Chosen-Ciphertext Security

Theorem D.4. *The above TREnc is TCCA-secure under the SXDH assumption and the collision resistance of the hash function. More precisely, we have* $\Pr[\mathsf{Exp}_{\mathcal{A},\Pi}^{\mathsf{tcca}}(\lambda) = 1] \leq \frac{1}{2} + \varepsilon_{\mathsf{cr}} + 2 \cdot \varepsilon_{\mathsf{sxdh}} + \Omega(2^{-\lambda})$.

Proof. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an efficient adversary against the TCCA security of our TREnc Π. We consider a sequence of games. In Game i, we denote by S_i the event that $\mathcal{A}$ wins by correctly guessing the internal random bit b of the game, which makes to game to output 1.

Game 0: This is the real game, where $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{Gen}(1^\lambda)$ with $\mathsf{SK} = (\alpha, \beta)$ and $\mathsf{PK} = (f, g, h, F, G, \mathsf{crs}_w, \varSigma_u, \varSigma_v, \mathsf{pk}$
Then, we input $\mathcal{A}_1$ with PK and we get back two ciphertexts CT_0 and CT_1. If $\mathsf{opk}_0 = \mathsf{Trace}(\mathsf{PK}, \mathsf{CT}_0) \neq \mathsf{Trace}(\mathsf{PK}, \mathsf{CT}_1) = \mathsf{opk}_1$ or $\mathsf{Ver}(\mathsf{PK}, \mathsf{CT}_0) \neq 1 \neq \mathsf{Ver}(\mathsf{PK}, \mathsf{CT}_1)$, abort the game and output a random bit. Otherwise, we pick a random bit $b \leftarrow\!\!\$\, \{0, 1\}$, and we return $\mathsf{CT}^* \leftarrow \mathsf{Rand}(\mathsf{PK}, \mathsf{CT}_b)$ to $\mathcal{A}_2$ which outputs its guess b'. If $b = b'$, we output 1, else, we output 0. We recall that $\mathcal{A}_1$ is allowed to make any (pre-challenge) decryption queries and that $\mathcal{A}_2$ is allowed to make any (post-challenge) decryption queries for any ciphertext that does not trace to $\mathsf{opk}^* := \mathsf{opk}_0 = \mathsf{opk}_1$. By definition, $\Pr[S_0] = \Pr[\mathsf{Exp}^{\mathsf{tcca}}_{\mathcal{A},\Pi}(\lambda) = 1]$.

Game 1: This game is as the real game except that we abort and output a random bit if the adversary produces two valid ciphertexts CT and CT' such that $H(\mathsf{opk}) = H(\mathsf{opk}')$ but $\mathsf{opk} \neq \mathsf{opk}'$.

Such an abort implies a collision on H. Thus, $|\Pr[S_0] - \Pr[S_1]| \leq \varepsilon_{\mathsf{cr}}$.

Game 2: In this game, we introduce a failure event with respect to the previous game which causes this game to abort and output a random bit. First of all, when we generate PK, we keep $\mathsf{sk}^{\mathsf{qazk}}_{\mathsf{lhsp}} := (\mathsf{sk}_u, \mathsf{sk}_v)$ associated to $\mathsf{pk}^{\mathsf{qazk}}_{\mathsf{lhsp}} = \{\hat{u}_1, \hat{u}_2, \hat{v}_1, \hat{v}_2\}$ that is used to compute $\varSigma_u$ and $\varSigma_v$ included in PK. Now, the failure event occurs if during the entire game the adversary produces a valid ciphertext $\mathsf{CT} = (\mathbf{c}, \boldsymbol{C}_Z, \boldsymbol{C}_R, \sigma_2, \sigma_3, \pi, \hat{\pi}_{\mathsf{sig}}, \mathsf{opk})$ at any time such that the one-time LHSP signature $\pi \neq \mathsf{Sign}(\mathsf{sk}^{\mathsf{qazk}}_{\mathsf{lhsp}}, (c_1^\tau, c_2^\tau, c_1, c_2))$, where $\boldsymbol{c} = (c_0, c_1, c_2)$ and $\tau = H(\mathsf{opk})$, except if $\mathsf{opk} = \mathsf{opk}^*$ in a post-challenge decryption query. So this can happen in any other case in a decryption queries or also during the challenge phase. Moreover, the failure event also occurs if during the challenge phase the given ciphertexts $\mathsf{CT}_i = (\mathbf{c}^{(i)}, \boldsymbol{C}_Z^{(i)}, \boldsymbol{C}_R^{(i)}, \sigma_2^{(i)}, \sigma_3^{(i)}, \pi^{(i)}, \hat{\pi}_{\mathsf{sig}}^{(i)}, \mathsf{opk}_i)$, for $i = 0, 1$, are valid but $(\sigma_2^{(0)}, \sigma_3^{(0)}) \neq (\sigma_2^{(1)}, \sigma_3^{(1)})$. Still, the generation of the challenge ciphertext computed by randomizing CT_b remains unchanged. This randomization preserves the (σ_2, σ_3)-signature components. Then, if the game does not abort we must have $\mathsf{opk}^* = \mathsf{opk}_0 = \mathsf{opk}_1$, $\sigma_2^* = \sigma_2^{(0)} = \sigma_2^{(1)}$ and $\sigma_3^* = \sigma_3^{(0)} = \sigma_3^{(1)}$. Note that we do not abort if a pre-challenge decryption query involves a valid ciphertext CT that trace to CT^*, i.e., $\mathsf{opk} = \mathsf{opk}^*$ but the (σ_2, σ_3)-signature components differ (as it will not be harmful).

Clearly, $|\Pr[S_1] - \Pr[S_2]|$ is bounded by the probability that $(\sigma_2^{(0)}, \sigma_3^{(0)}) \neq (\sigma_2^{(1)}, \sigma_3^{(1)})$ occurs or that at least one signature π of a valid ciphertext is not the honest one that would be computed with $\mathsf{sk}^{\mathsf{qazk}}_{\mathsf{lhsp}}$. Given the public parameters $\hat{g}, \hat{h}$ of the one-time LHSP signature scheme, we can build a straightforward reduction to DP if we have any 2 distinct valid one-time LHSP signatures on the same vector. We thus have, $|\Pr[S_1] - \Pr[S_2]| \leq \varepsilon_{\mathsf{sxdh}}$.

Game 3: This game is as Game 2 except in the way we generate the challenge ciphertext by randomization. First, when we generate PK, we keep $\mathsf{sk}^{\mathsf{qazk}}_{\mathsf{lhsp}} := (\mathsf{sk}_u, \mathsf{sk}_v)$ as before. Second, when we create the encryption CT^* by randomizing CT_b, we simulate the proof π^* by re-signing (c_1^*, c_2^*) from scratch using the secret key $\mathsf{sk}^{\mathsf{qazk}}_{\mathsf{lhsp}}$ at Step 3. That is, when we randomize CT_b by picking $\theta^* \leftarrow\!\!\$\, \mathbb{Z}_p$ and first computing $\boldsymbol{c}^* = \boldsymbol{c}^{(b)} \cdot (f, g, h)^{\theta^*}$ so that $\boldsymbol{c}^* =: (c_0^*, c_1^*, c_2^*) =$

$(c_0^{(b)} \cdot f^{\theta^*}, c_1^{(b)} \cdot g^{\theta^*}, c_2^{(b)} \cdot h^{\theta^*})$, we no more compute the proof as $\pi^* = \pi^{(b)} \cdot (\Sigma_u^{\tau_b} \Sigma_v)^{\theta^*}$, where $\tau_b = H(\mathsf{opk}_b)$, as Rand would do.

Game 2 and Game 3 abort the game in the same cases. Moreover, when they do not abort, the view in both games are actually exactly the same. Indeed, since $\pi^{(b)} = \mathsf{Sign}(\mathsf{sk}_{\mathsf{lhsp}}^{\mathsf{qazk}}, (c_1^{(b)\tau_b}, c_2^{(b)\tau_b}, c_1^{(b)}, c_2^{(b)}))$ when there is no abort, we have $\pi^{(b)} \cdot (\Sigma_u^{\tau_b} \Sigma_v)^{\theta^*} = \mathsf{Sign}(\mathsf{sk}_{\mathsf{lhsp}}^{\mathsf{qazk}}, (c_1^{*\tau_b}, c_2^{*\tau_b}, c_1^*, c_2^*))$ by the homomorphic property of the deterministic one-time LHSP signature scheme since $(\Sigma_u^{\tau_b} \Sigma_v)^{\theta^*} = \mathsf{Sign}(\mathsf{sk}_{\mathsf{lhsp}}^{\mathsf{qazk}}, (g^{\theta^* \tau_b}, h^{\theta^* \tau_b}, c_1^{\theta^*}, c_2^{\theta^*}))$. We thus have $\Pr[S_2] = \Pr[S_3]$.

Game 4: In this game we bring yet another modification in the way we generate the challenge ciphertext CT^* by randomizing CT_b with respect to the previous game. During the generation of the public key, we keep the value H such that $(F, G, H) = (f, g, h)^\delta$. At Step 1 of the randomization algorithm we now compute the CPA encryption part as $\boldsymbol{c}^* = \boldsymbol{c}^{(b)} \cdot (f, g, h)^{\theta^*} \cdot (F, G, H)^{\rho^*}$, for random $\theta^*, \rho^* \leftarrow\!\$\, \mathbb{Z}_p$. Then, we also have to modify the way we derive a committed valid one-time LHSP signature on (g, c^*, c_1^*) at Step 2. We now compute $\tilde{\sigma}_1^* := (\tilde{Z}_1^*, \tilde{R}_1^*) = (Z_2^{(b)\theta^*} \cdot Z_3^{(b)\rho^*}, R_2^{(b)\theta^*} \cdot R_3^{(b)\rho^*}) = \sigma_2^{(b)\theta^*} \cdot \sigma_3^{(b)\rho^*}$, which consists of a one-time LHSP signature on $(1, f, g)^{\theta^*} \cdot (F, G, H)^{\rho^*}$. The remaining part of that step is done as in Rand. That is, we adapt and randomize the commitments $\boldsymbol{C}_Z^* = \boldsymbol{C}_Z^{(b)} \cdot \iota(\tilde{Z}_1^*) \vec{w}_1^{z_1^*} \vec{w}_2^{z_2^*}$ and $\boldsymbol{C}_R^* = \boldsymbol{C}_R^{(b)} \cdot \iota(\tilde{R}_1^*) \vec{w}_1^{r_1^*} \vec{w}_2^{r_2^*}$, for some random scalars $z_1^*, z_2^*, r_1^*, r_2^* \leftarrow\!\$\, \mathbb{Z}_p$, as well as the proof so that $\hat{\pi}_{\mathsf{sig}}^* := (\hat{P}_1^*, \hat{P}_2^*) = (\hat{P}_1^{(b)} \cdot \hat{g}^{z_1^*} \hat{h}^{r_1^*}, \hat{P}_2^{(b)} \cdot \hat{g}^{z_2^*} \hat{h}^{r_2^*})$, where $\hat{\pi}_{\mathsf{sig}}^{(b)} = (\hat{P}_1^{(b)}, \hat{P}_2^{(b)})$.

Since $\boldsymbol{c}^* = \boldsymbol{c}^{(b)} \cdot (f, g, h)^{\theta^* + \delta\rho^*}$, the CPA encryption part is randomized exactly as in Game 3 as well as the simulated proof $\pi^* = \mathsf{Sign}(\mathsf{sk}_{\mathsf{lhsp}}^{\mathsf{qazk}}, (c_1^{*\tau_b}, c_2^{*\tau_b}, c_1^*, c_2^*))$, where $\tau^* = H(\mathsf{opk}^*)$. Moreover, the Groth-Sahai commitments and proofs are fully randomized and redistributed among all the valid commitments and proofs that satisfy the verification equation with the constants $g, c_0^*, c_1^* \in \mathbb{G}$ and $\hat{g}, \hat{h} \in \hat{\mathbb{G}}$, and $\mathsf{opk}^* = \mathsf{opk}_b$. Indeed, since CT_b contains a valid signature $\sigma_3^{(b)}$ on the vector $(1, F, G)$ and a valid Groth-Sahai commitments-and-proof related to opk_b if the game does not abort (and it aborts in the same way in Game 3 and Game 4), the same holds for CT^* which has thus exactly in the same distribution in both games. We have $\Pr[S_3] = \Pr[S_4]$.

Game 5: In this game, we change the way we compute (F, G, H) during the generation of the public key. Now, we pick random $G, H \leftarrow\!\$\, \mathbb{G}$ and compute $F = G^\alpha H^\beta$ using the secret key $\mathsf{SK} = (\alpha, \beta)$. We recall that $f = g^\alpha h^\beta$ and that H is not included in PK. The remaining part of the public key is unchanged as well as the way we deal with decryption queries and the computation of the challenge ciphertext $\mathsf{CT}^* = (\mathbf{c}^*, \boldsymbol{C}_Z^*, \boldsymbol{C}_R^*, \sigma_2^*, \sigma_3^*, \pi^*, \hat{\pi}_{\mathsf{sig}}^*, \mathsf{opk}^*)$ by randomization. Still, as a side effect $\boldsymbol{c}^* = (c_0^*, c_1^*, c_2^*)$ is no more in the range of the honest CPA encryptions of $\mathsf{Dec}(\mathsf{SK}, \mathsf{CT}_b)$ except with probability $1/p$. This also implies that π^* is a valid proof of a false statement.

Obviously, we have $|\Pr[S_4] - \Pr[S_5]| \leq \varepsilon_{\mathsf{sxdh}}$ since the only distinction between Game 4 and Game 5 is that $(g, h, G, H) \in \mathbb{G}^4$ is a random DH tuple in the former game and random quadruple in the latter game.

Game 6: In this game, we still generate $(g, h, G, H) \in \mathbb{G}^4$ uniformly at random but we compute $h = g^\gamma$ for a known random scalar $\gamma \leftarrow\!\$\, \mathbb{Z}_p$. Then, we introduce one more failure event which causes the game to abort and output a random bit. This event occurs if the adversary manages to compute a ciphertext CT that it is deemed valid in the previous game but $c_1^\gamma \neq c_2$ if at least one of the following situations: in any pre-challenge decryption query, in the challenge phase with CT_0 or CT_1, or in any post-challenge decryption query with $\mathsf{Trace}(\mathsf{PK}, \mathsf{CT}) \neq \mathsf{opk}^*$. In other words, we reject all the valid ciphertexts in the sense of Game 4 for which π is a valid proof for $\boldsymbol{c}$ while this CPA ciphertext is not in the range of the honest encryptions of $\mathsf{Dec}(\mathsf{SK}, \mathsf{CT})$, and γ is a trapdoor membership key that allows us to figure out whether π is actually a fake proof.

It is clear that $|\Pr[S_5] - \Pr[S_6]|$ is bounded by the probability that the new failure event occurs. From Game 2, and the successive additional abort rules, a valid ciphertext $\mathsf{CT} = (\mathbf{c}, \boldsymbol{C}_Z, \boldsymbol{C}_R, \sigma_2, \sigma_3, \pi, \hat{\pi}_{\mathsf{sig}}, \mathsf{opk})$ that is not rejected satisfies $\pi = \mathsf{Sign}(\mathsf{sk}^{\mathsf{qazk}}_{\mathsf{lhsp}}, (c_1^\tau, c_2^\tau, c_1, c_2))$, where $\boldsymbol{c} = (c_0, c_1, c_2)$ and $\tau = H(\mathsf{opk})$. Now, we show that all such ciphertexts must also satisfy $c_1^\gamma = c_2$, except with negligible probability $(q+2)/(p-q-2)$.
Indeed, let us consider what a computationally unbounded adversary $\mathcal{A}$ can infer $\mathsf{sk}^{\mathsf{qazk}}_{\mathsf{lhsp}} =:= \{(\mu_1, \mu_2, \mu_3, \mu_4, \nu_1, \nu_2, \nu_3, \nu_4)\}$ during the whole game, where $\mathsf{sk}_u =: \{(\mu_1, \nu_1), (\mu_2, \nu_2)\}$ and $\mathsf{sk}_v =: \{(\mu_3, \nu_3), (\mu_4, \nu_4)\}$. In the public key $\mathsf{pklhsp}^{\mathsf{qazk}}$, the discrete logarithms of $\{\hat{u}_i = \hat{g}^{\mu_i} \cdot \hat{h}^{\nu_i}, \hat{v}_i = \hat{g}^{\mu_{i+2}} \cdot \hat{h}^{\nu_{i+2}}, \}_{i=1}^2$ provide 4 linear equations and those of $\{(Z_u, R_u), (Z_v, R_v)\}$ included in PK provide $\mathcal{A}$ with two additional independent linear equations. In addition, $\{R_u, R_v\}$ are uniquely determined $\{Z_u, Z_v\}$ and do not reveal any more information than them. As a consequence, from $\mathcal{A}$'s point view, the vector $(\mu_1, \mu_2, \mu_3, \mu_4, \nu_1, \nu_2, \nu_3, \nu_4)$ is uniformly distributed in a two-dimensional subspace at the beginning of the game. Hence, at the first pre-challenge decryption query such that $c_1^\gamma = c_2$ and then $(c_1^\tau, c_2^\tau, c_1, c_2)$ is not in the span generated by the vectors $(g, h, 1, 1)$ and $(1, 1, g, h)$ for which we gave the signatures $\varSigma_u = (Z_u, R_u)$ and $\varSigma_v = (Z_v, R_v)$, the equalities

$$Z = c_1^{\tau\mu_1} \cdot c_2^{\tau\mu_2} \cdot c_1^{\mu_3} \cdot c_2^{\mu_4}, \qquad\qquad R = c_1^{\tau\nu_1} \cdot c_2^{\tau\nu_2} \cdot c_1^{\nu_3} \cdot c_2^{\nu_4} \tag{5}$$

can only hold with probability $1/p$. However, each query potentially allows $\mathcal{A}$ to eliminate one candidate for the vector $(\mu_1, \mu_2, \mu_3, \mu_4, \nu_1, \nu_2, \nu_3, \nu_4)$. At the k-th pre-challenge query, the equalities (5) thus hold with probability smaller than $1/(p-k)$. After the challenge phase, since $c_1^{*\gamma} \neq c_2^*$ with probability 1/p, we give one more independent linear equation about $\mathsf{sk}^{\mathsf{qazk}}_{\mathsf{lhsp}}$ and it still remains one-degree of freedom. Now, in the next valid post-challenge queries with $\tau = H(\mathsf{opk}) \neq H(\mathsf{opk}^*) = \tau^*$ assuming that we do not abort (we relied on the collision resistance in Game 1), we will also have a probability smaller than $1/(p - q_1 - 2 - k)$ to fulfill the equalities (5) at the k-th post-challenge query, where q_1 is the number of pre-challenge queries and the 2 counts for CT_0 and CT_1. Indeed, if we write $\boldsymbol{c} = (c_1^\tau, c_2^\tau, c_1, c_2) = (g^{\vartheta_1\tau}, g^{\gamma\vartheta_2\tau}, g^{\vartheta_1}, g^{\gamma\vartheta_2})$ and $\boldsymbol{c}^* = (c_1^{*\tau^*}, c_2^{*\tau^*}, c_1^*, c_2^*) = (g^{\theta_1\tau^*}, g^{\gamma\theta_2\tau^*}, g^{\theta_1}, g^{\gamma\theta_2})$ for some $\vartheta_1 \neq \vartheta_2$ and $\theta_1 \neq \theta_2$, we see that

$$\det \begin{pmatrix} 1 & \gamma & 0 & 0 \\ 0 & 0 & 1 & \gamma \\ \theta_1\tau^* & \gamma\theta_2\tau^* & \theta_1 & \gamma\theta_2 \\ \vartheta_1\tau & \gamma\vartheta_2\tau & \vartheta_1 & \gamma\vartheta_2 \end{pmatrix} = \gamma^2(\tau^* - \tau)(\theta_1 - \theta_2)(\vartheta_1 - \vartheta_2).$$

where the first two rows represent $(g, h, 1, 1)$ and $(1, 1, g, g)$. The inequality $|\Pr[S_5] - \Pr[S_6]| \leq (q+2)/(p-q-2)$ follows.

To conclude, we argue that $\mathcal{A}$'s view in Game 6 is statistically independent of the hidden bit b. If the game aborts and outputs a random bit, the probability to return 1 is $1/2$. So, let us evaluate the probability that $\mathcal{A}$ returns b at the end of the game when there is no abort. Since there is no abort, all the ciphertexts CT for which we compute $\mathsf{Dec}(\mathsf{SK}, \mathsf{CT}) = (c_0 \cdot c_1^{-\alpha} \cdot c_2^{-\beta})$ does not reveal any additional information about the secret key $\mathsf{SK} = (\alpha, \beta)$ beyond what can be inferred from $f = g^\alpha h^\beta$ and $F = G^\alpha H^\beta$ which thus remain independent since H is kept secret until the computation of the challenge ciphertext CT^*. Let us write $G = g^x$ and $H = h^x f^z$, for random $x, z \in \mathbb{Z}_p$, so that $F = f^{x+\beta z}$. The computation of $\boldsymbol{c}^* = \boldsymbol{c}^{(b)} \cdot (f, g, h)^{\theta^*} \cdot (F, G, H)^{\rho^*}$, for random $\theta^*, \rho^* \leftarrow\!\$\, \mathbb{Z}_p$, introduced in Game 4 implies that $c_0^* = c_0^{(b)} \cdot f^{\theta^* + \rho^*(x+\beta z)}$, $c_1^* = c_1^{(b)} \cdot g^{\theta^* + \rho^* x}$, and $c_2^* = c_2^{(b)} \cdot h^{\theta^* + \rho^* x} \cdot f^{\rho^* z}$. Therefore, as long as $x, z, \rho^* \neq 0$, $\boldsymbol{c}^*$ is a random triple over $\mathbb{G}^3$ since H is still not added to $\mathcal{A}$'s view. Hence, $\Pr[S_6] = 1/2 + 3/p$.

In summary, we find

$$\Pr[\mathsf{Exp}^{\mathrm{tcca}}_{\mathcal{A},\Pi}(\lambda) = 1] \leq \frac{1}{2} + \varepsilon_{\mathsf{cr}} + 2 \cdot \varepsilon_{\mathsf{sxdh}} + \frac{q+2}{p-q-2} + \frac{3}{p}.$$

□

E Asymmetric Waters Signature

We define the Waters signature scheme for messages $m = (m_1, \ldots, m_k) \in \{0,1\}^k$.

Setup(1^λ): Generate $(p, \mathbb{G}, \hat{\mathbb{G}}, g, \hat{g}, \mathbb{G}_T, e)$ as in Sec. 4.1, pick $z \leftarrow\!\$\, \mathbb{G}$ as well as $\mathbf{u} = (u_0, \ldots, u_k) \leftarrow\!\$\, \mathbb{G}^{k+1}$. Output $pp = (p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e, g, \hat{g}, z, \mathbf{u})$.

SKeyGen(pp): Choose $x \leftarrow\!\$\, \mathbb{Z}_p$, define $X = g^x, \hat{X} = \hat{g}^x, Y = z^x$; output the public key $vk = (pp, X, \hat{X})$ and the secret key $sk = (pp, Y)$.

Sign($sk, m; s$): Define $\mathcal{F}(m) := u_0 \Pi_{i=1}^k u_i^{m_i}$. For randomness $s \in \mathcal{Z}_p$, return the signature σ defined as:

$$(\sigma_1 = Y \cdot \mathcal{F}(m)^s, \sigma_2 = g^s, \sigma_3 = \hat{g}^s)$$

Verif(vk, m, σ): Output 1 if both of the following hold and 0 otherwise:

$$e(\sigma_1, \hat{g}) = e(z, \hat{X}) \cdot e(\mathcal{F}(m), \sigma_3) \text{ and } e(\sigma_2, \hat{g}) = e(g, \sigma_3)$$

Random($vk, F, \sigma; s'$): For $F = \mathcal{F}(m)$ and randomness $s' \in \mathcal{Z}_p$, output $\sigma' = (\sigma_1 \cdot F^{s'}, \sigma_2 \cdot g^{s'}, \sigma_3 \cdot \hat{g}^{s'})$

F Belenios RF model

As our receipt-freeness definition for a voting scheme is inspired from BeleniosRF[14], we find it useful to remind their model. The main difference with our model is the following: in addition to the parties used in our model, there is an additional party, the registrar, which provides to the users voting credentials ($\mathsf{upk}_{\mathsf{id}}, \mathsf{usk}_{\mathsf{id}}$) when they register to the election. Voters make use of these credentials to craft a valid ballot and verify that their ballot is included in the tally. Even though there is no explicit $\mathsf{TraceBallot}$ algorithm, the public credentials upk can be used to trace ballots computed with the associated usk.

Definition F.1 (Voting System). *A Voting System is a tuple of probabilistic polynomial-time algorithms* ($\mathsf{SetupElection}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Valid}, \mathsf{Append}, \mathsf{Publish}, \mathsf{VerifyVote}, \mathsf{Tally}, \mathsf{VerifyResult}$) *associated to a result function* $\rho_m : \mathbb{V}^m \cup \{\perp\} \to \mathbb{R}$ *where* $\mathbb{V}$ *is the set of valid votes and* $\mathbb{R}$ *is the result space such that:*

- $\mathsf{SetupElection}(1^\lambda)$*: on input security parameter* 1^λ*, generate the public and secret keys* $(\mathsf{pk}, \mathsf{sk})$ *of the election.*
- $\mathsf{Register}(\mathsf{id})$*: on input an identifier* id*, outputs the secret part of the credential* $\mathsf{usk_{id}}$ *and its public credential* $\mathsf{upk_{id}}$*, which is added to the list* $L = \{\mathsf{upk_{id}}\}$
- $\mathsf{Vote}(\mathsf{v}, \mathsf{upk}, \mathsf{usk}, \mathsf{v})$*: is run by a voter* id*with credentials* upk*,* usk *to cast their vote* $\mathsf{v} \in \mathbb{V}$*. It outputs a ballot* b*, which is sent to the voting server (possibly through an authenticated channel).*
- $\mathsf{Valid}(\mathsf{BB}, \mathsf{b})$ *takes as input the ballot box* BB *and a ballot* b *and checks the validity of the latter. Outputs* 1 *if the ballot is valid and* 0 *otherwise (e.g. ill-formed, containing duplicated ciphertext from the ballot box...).*
- $\mathsf{Append}(\mathsf{BB}, \mathsf{b})$ *updates* BB *with the ballot* b*. Typically, this consists in adding* b *as a new entry to* BB*, but more involved actions might be possible.*
- $\mathsf{Publish}(\mathsf{BB})$*: on input ballot box* BB*, outputs the public view* PBB *of* BB*.*
- $\mathsf{VerifyVote}(\mathsf{PBB}, \mathsf{id}, \mathsf{upk}, \mathsf{usk}, \mathsf{b})$ *is run by voters for checking that their ballots will be included in the tally. On inputs the public board* PBB*, a ballot* b*, and the voter's identity and credentials* $\mathsf{id}, \mathsf{uskupk}$*, it returns* 1 *or* 0*.*
- $\mathsf{Tally}(\mathsf{BB}, \mathsf{sk})$*: on input ballot box* BB *and private key of the election* sk*, outputs the tally* r *(consistent with the result function* ρ *applied on the votes contained in the ballots of* BB*) and a proof* Π *of correct tabulation.*
- $\mathsf{VerifyResult}(\mathsf{PBB}, \mathsf{r}, \Pi)$*: on input public ballot box* PBB*, result of the tally* r *and proof of the tally* Π*, checks that the result of the tally is correct with regard to* PBB *and the associated proof. Outputs* 1 *if the tally is valid and* 0 *otherwise.*

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

Regarding the receipt-freeness definition, the main difference is the absence of the trace check in the $\mathcal{O}\mathsf{receiptLR}$ oracle. However, this check is implicitly done in the Valid algorithm: if b_0 and b_1 are associated to different credentials, then the bulletin boards will be distinguishable. Hence, Valid has to somehow verify that the ballot is indeed corresponding to the voter that sent it! These credentials helps to achieve receipt-freeness, however the voting system rests on the honesty of the registrar producing these credentials. Indeed, a corrupted registrar can vote in any voter's place without any way to detect it.

Definition F.2 (Receipt-Freeness). *Let* $\mathcal{V}$ *be a voting protocol for a set ID of voter identities and a result function* ρ*. We say that* $\mathcal{V}$ *has strong receipt-freeness if there exist algorithms* $\mathsf{SimSetup}$ *and* $\mathsf{SimProof}$ *such that no efficient adversary can distinguish between games* $\mathsf{Exp}^{\mathsf{srf},0}_{\mathcal{A},\mathcal{V}}(\lambda)$ *and* $\mathsf{Exp}^{\mathsf{srf},0}_{\mathcal{A},\mathcal{V}}(\lambda)$ *defined by the oracles in Figure 8; that is, for any efficient algorithm* $\mathcal{A}$ *the following is negligible in* λ*:*

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

is negligible in λ.

```
𝒪init(λ)
    if $\beta = 0$ then $(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{SetupElection}(1^\lambda)$
    else $(\mathsf{pk}, \mathsf{sk}, \tau) \leftarrow\$ \mathsf{SimSetup}(1^\lambda)$
    return pk

𝒪voteLR(id, v0, v1)
    if $\mathsf{v}_0 \notin \mathbb{V}$ or $\mathsf{v}_1 \notin \mathbb{V}$ then return $\perp$
    $\mathsf{b}_0 = \mathsf{Vote}(\mathsf{id}, \mathsf{upk}_{\mathsf{id}}, \mathsf{usk}_{\mathsf{id}}, \mathsf{v}_0)$
    $\mathsf{b}_1 = \mathsf{Vote}(\mathsf{id}, \mathsf{upk}_{\mathsf{id}}, \mathsf{usk}_{\mathsf{id}}, \mathsf{v}_1)$
    $\mathsf{Append}(\mathsf{BB}_0, \mathsf{b}_0); \mathsf{Append}(\mathsf{BB}_1, \mathsf{b}_1)$

𝒪reg(id)
    if id was not previously queried,
    then run Register(id) and set,
    $\mathcal{U} = \mathcal{U} \cup \{(\mathsf{id}, \mathsf{upk}_{\mathsf{id}}, \mathsf{usk}_{\mathsf{id}})\}$
    return $\mathsf{upk}_{\mathsf{id}}$

𝒪receiptLR(id, b0, b1)
    if $id \notin \mathcal{CU}$ then return $\perp$
    if $\mathsf{Valid}(\mathsf{BB}_0, \mathsf{b}_0) = 0$ or $\mathsf{Valid}(\mathsf{BB}_1, \mathsf{b}_1) = 0$
    then return $\perp$
    else $\mathsf{Append}(\mathsf{BB}_0, \mathsf{b}_0); \mathsf{Append}(\mathsf{BB}_1, \mathsf{b}_1)$

𝒪corruptU(id)
    On a registered voter id, output $(\mathsf{upk}_{\mathsf{id}})$
    Set $\mathcal{CU} = \mathcal{CU} \cup \{(id, \mathsf{upk}_{\mathsf{id}})\}$

𝒪board()
    return $\mathsf{Publish}(\mathsf{BB}_\beta)$

𝒪cast(id, b)
    if $\mathsf{Valid}(\mathsf{BB}_\beta, \mathsf{b}) = 0$ then return $\perp$
    else $\mathsf{Append}(\mathsf{BB}_0, \mathsf{b}); \mathsf{Append}(\mathsf{BB}_1, \mathsf{b})$

𝒪tally()
    $(\mathsf{r}, \Pi) \leftarrow\$ \mathsf{Tally}(\mathsf{BB}_0, \mathsf{sk})$
    if $\beta = 1$ then $\Pi \leftarrow\$ \mathsf{SimProof}(\mathsf{BB}_1, \mathsf{r}, \tau)$
    return $(\mathsf{r}, \Pi)$
```

Fig. 8. Oracles used in the strong receipt-freeness games of the BeleniosRF model. The $\beta = 0$ game corresponds to the real ballot box, while the $\beta = 1$ game correspond to the fake ballot box

G Ballot privacy

Receipt-freeness can naturally be seen as a stronger form of ballot privacy as replacing the $\mathcal{O}\mathsf{receiptLR}$ oracle in the receipt-freeness game with the $\mathcal{O}\mathsf{voteLR}$ and $\mathcal{O}\mathsf{cast}$ of Figure 9 yields the BPRIV game [8], and those two oracles can be simulated by $\mathcal{O}\mathsf{receiptLR}$.

```
𝒪voteLR(id, v0, v1)
    if $\mathsf{v}_0 \notin \mathbb{V}$ or $\mathsf{v}_1 \notin \mathbb{V}$ then return $\perp$
    $\mathsf{b}_0, \mathsf{aux} = \mathsf{Vote}(\mathsf{id}, \mathsf{v}_0)$
    $\mathsf{b}_1 = \mathsf{Vote}(\mathsf{id}, \mathsf{v}_1, \mathsf{aux})$
    $\mathsf{Append}(\mathsf{BB}_0, \mathsf{b}_0); \mathsf{Append}(\mathsf{BB}_1, \mathsf{b}_1)$

𝒪cast(b)
    if $\mathsf{Valid}(\mathsf{BB}_\beta, \mathsf{b}) = 0$ then return $\perp$
    else $\mathsf{Append}'(\mathsf{BB}_0, \mathsf{b}); \mathsf{Append}'(\mathsf{BB}_1, \mathsf{b})$
```

Fig. 9. Additional oracles used to transform the receipt-freeness game into the BPRIV game. Both $\mathcal{O}\mathsf{voteLR}$ and $\mathcal{O}\mathsf{cast}$ can be simulated by $\mathcal{O}\mathsf{receiptLR}$.

When BPRIV is supplemented by two additional security properties, strong consistency and strong correctness, it has been shown that for a voting scheme secure in this regard, an adversary can extract as much information as it can extract from only seeing the result and nothing more.

Definition G.1 (Strong consistency). *A scheme $\mathcal{V}$ has strong consistency if there an extraction algorithm* $\mathsf{Extract}$ *that takes as input a secret key* sk *and a ballot* b *and outputs* $\mathsf{v} \in \mathbb{V}$ *which satisfies these conditions:*

- *For any* $(\mathsf{pk}, \mathsf{sk})$ *in the range of* $\mathsf{SetupElection}$*, for any vote* v *and any* aux*, if* $\mathsf{b}, \mathsf{aux} \leftarrow\$ \mathsf{Vote}(\mathsf{id}, \mathsf{v})$ *then* $\mathsf{Extract}(\mathsf{sk}, \mathsf{b}) = \mathsf{v}$ *with overwhelming probability.*
- *Consider an adversary $\mathcal{A}$ which is given* pk *and consider the experiment* $\mathsf{Exp}^{\text{strong\_consistency}}_{\mathcal{A},\mathcal{V}}(\lambda)$ *given in Figure 10. We require that the probability* $Pr[\mathsf{Exp}^{\text{strong\_consistency}}_{\mathcal{A},\mathcal{V}}(\lambda) = 1]$ *is negligible in* λ*.*

```
$\mathsf{Exp}^{\text{strong\_consistency}}_{\mathcal{A},\mathcal{V}}(\lambda)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{SetupElection}(1^\lambda)$
$\mathsf{BB} \leftarrow\$ \mathcal{A}(\mathsf{pk})$
$(\mathsf{r}, \Pi) \leftarrow\$ \mathsf{Tally}(\mathsf{BB}, \mathsf{sk})$
if $r \neq \rho(\mathsf{Extract}(\mathsf{sk}, \mathsf{b}_1), \ldots, \mathsf{Extract}(\mathsf{sk}, \mathsf{b}_n))$
then return $1$else return $0$;
```

Fig. 10. Strong consistency experiment. We consider an adversary $\mathcal{A}$ that returns BB of the form $[\mathsf{b}_1, \ldots, \mathsf{b}_n]$ such that $\mathsf{Valid}(\mathsf{BB}, \mathsf{b}_i) = 1$ for $i = 1, \ldots, n$.

Definition G.2 (Strong correctness). *A scheme $\mathcal{V}$ has strong correctness if for any adversary $\mathcal{A}$ that takes as input* pk*, the probability*

$$Pr[(\mathsf{id}, \mathsf{v}, \mathsf{BB}) \leftarrow\$ \mathcal{A}(\mathsf{pk}); \mathsf{b}, \mathsf{aux} \leftarrow\$ \mathsf{Vote}(\mathsf{id}, \mathsf{v}); \mathsf{Valid}(\mathsf{BB}, b) = 0]$$

is negligible in λ*.*

The protocol we propose in Section 5 has strong consistency and strong correctness. The strong consistency comes from the consistency of the tallying algorithm. Regarding the strong correctness, the ballot $\mathsf{b} \leftarrow\$ \mathsf{Vote}(\mathsf{id}, \mathsf{v})$ of any voter id and any vote v is not valid only if

- $\mathsf{Ver}(b) = 0$, which can only happen with a negligible probability on λ because of the honest verifiability of the TREnc (Definition 2.2)
- There is already a ballot on BB that has the same trace, which can only happen with a negligible probability because of the traceability property.

Hence, the scheme has ballot privacy against every party except the rerandomizing server, who is supposed to be honest in the receipt-freeness game! Now if the rerandomizing server is corrupted (in the honest-but-curious model), we have to slightly modify the BPRIV game since it has an additional view of the election (it sees the ballots sent by the voters before the $\mathsf{ProcessBallot}$ algorithm), In the following alternative game, the $\mathcal{O}\mathsf{voteLR}$ algorithm returns b_β, the ballot sent by the voter which will end up in the adversary's ballot box after $\mathsf{ProcessBallot}$.

Definition G.3 (Ballot privacy). *A voting system $\mathcal{V}$ has ballot privacy against the rerandomizing server if there exists algorithms* $\mathsf{SimProof}$ *and* $\mathsf{SimSetupElection}$ *such that no* PPT *adversary $\mathcal{A}$ can distinguish between games $\mathsf{Exp}^{\mathsf{bpriv},0}_{\mathcal{A},\mathcal{V}}(\lambda)$ and $\mathsf{Exp}^{\mathsf{bpriv},1}_{\mathcal{A},\mathcal{V}}(\lambda)$ defined by the oracles in Figure 11, that is for any efficient algorithm $\mathcal{A}$:*

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

is negligible in λ.

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

```
$\mathcal{O}\mathsf{voteLR}(\mathsf{id},\mathsf{v}_0,\mathsf{v}_1)$
if $\mathsf{v}_0 \notin \mathbb{V}$ or $\mathsf{v}_1 \notin \mathbb{V}$ then return $\perp$
$\mathsf{b}_0, \mathsf{aux} = \mathsf{Vote}(\mathsf{id},\mathsf{v}_0)$
$\mathsf{b}_1 = \mathsf{Vote}(\mathsf{id},\mathsf{v}_1,\mathsf{aux})$
$\mathsf{Append}'(\mathsf{BB}_0,\mathsf{b}_0); \mathsf{Append}'(\mathsf{BB}_1,\mathsf{b}_1)$
return $b_\beta$
```

```
$\mathcal{O}\mathsf{cast}(\mathsf{b})$
if $\mathsf{Valid}(\mathsf{BB}_\beta,\mathsf{b}) = 0$ then return $\perp$
else $\mathsf{Append}'(\mathsf{BB}_0,\mathsf{b}); \mathsf{Append}'(\mathsf{BB}_1,\mathsf{b})$
```

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

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

Fig. 11. Oracles used in the ballot privacy against the rerandomizing server games. The $\beta = 0$ game corresponds to the real ballot box, while the $\beta = 1$ game correspond to the fake ballot box. Except the lack of the $\mathcal{O}\mathsf{receiptLR}$ oracle and the addition of the $\mathcal{O}\mathsf{voteLR}$ and $\mathcal{O}\mathsf{cast}$ oracles, the only other differences with the receipt-freeness oracles of Figure 3 are that the $\mathcal{O}\mathsf{board}$ oracle returns BB instead of $\mathsf{Publish}(\mathsf{BB})$ and Append is replaced by Append' which runs Append and returns the randomness used in the execution. Here, since the rerandomizing server is corrupted, the $\mathcal{O}\mathsf{voteLR}$ oracle returns to the adversary the ballot that will end up in the adversary's ballot box after $\mathsf{ProcessBallot}$.

Indeed, nothing prevents so far the Vote algorithm to return the plaintext of the vote and the $\mathsf{ProcessBallot}$ algorithm to provide all the randomness necessary to hide the vote on PBB. This clearly offers no privacy against the rerandomizing server and we thus want a property analogous to the strong randomizability of a TREnc (see Definition 2.6). To achieve ballot privacy against the rerandomizing server, we need the following property:

Definition G.4 (Strong validity). *A voting scheme $\mathcal{V}$ has strong validity if:*

- *For every* pk *in the range of* $\mathsf{SetupElection}(1^\lambda)$*, every ballot box* BB *and every ballot* b *such that* $\mathsf{Valid}(\mathsf{BB},\mathsf{b}) = 1$*, we have that* b *is in the range of* $\mathsf{Vote}(\cdot)$*.*

– *For every* pk *in the range of* $\mathsf{SetupElection}(1^\lambda)$ *and every* b, aux *in the range of* $\mathsf{Vote}(\mathsf{id}, \mathsf{v})$*, the following computational indistinguishability relation holds:*

$$\{\mathsf{ProcessBallot}(\mathsf{b})\} \approx_c \{\mathsf{Vote}(\mathsf{id}, \mathsf{v}, \mathsf{aux})\}$$

We only sketch the proof here, which is similar to the receipt-freeness proof. Like the receipt-freeness proof, we also simulate the proofs of the tally if $\beta = 0$ in the first transition. Then, we add another transition that modify the $\mathcal{O}\mathsf{voteLR}$ in the following way. Instead of running $\mathsf{ProcessBallot}$ on b_β in the Append' procedure, b_β is directly appended to BB_β and the oracle return $\mathsf{ProcessBallot}(\mathsf{b}_\beta)$. This is indistinguishable from a normal $\mathcal{O}\mathsf{voteLR}$ execution (given in Figure 9) because of the strong validity and the rerandomizing server has now the same view as anyone else in the BPRIV game.

The protocol we propose satisfies strong validity. Indeed, the two items are respectively implied by the verifiability security (see 2.3) and the strong randomizability (see 2.6) of the TREnc. Hence our protocol offers ballot privacy against the rerandomizing server.

H The need of strong validity

In this section we provide a modified version of our practical SXDH-based TREnc of Section 4. This new scheme is still TCCA secure but it only satisfies the correctness requirement of *publicly traceable randomization* statistically (and, not perfectly) and for honestly computed ciphertext $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$ (rather than all the ciphertexts in the range of $\mathsf{Enc}(\mathsf{pk}, m)$), but still for any message m. Therefore, this new scheme is not a TREnc. Moreover, it also only satisfies a relaxed notion of *strong randomization* (Definition 2.6), where indistinguishability holds for any message m and link key lk, but only when ciphertexts are generated honestly.

However, this scheme still enjoys properties that are sufficient to realize a strong receipt-free e-voting in the sense of the BeleniosRF receipt-freeness definition, recalled in Appendix F.2 (and modified with the extra requirement that $\mathcal{O}\mathsf{receiptLR}$ checks that b_0 and b_1 are signed with the same key, that is, have identical traces in our setting, which is needed if we do not want to make this game trivial to win). Still, we will observe that this scheme is obviously not receipt-free, in the sense that a voter submitting a carefully chosen ciphertext as his ballot will be able to prove how he voted to any third-party.

This stresses that the BeleniosRF definition of receipt-freeness cannot be meaningful alone: we also need the voting scheme to satisfy our strong validity property and, indeed, a voting scheme built from our modified encryption scheme won't be strongly valid.

The idea behind this scheme is that it might exist valid ciphertexts, in the sense of Definition 2.3 (i.e., that are in the range of $\mathsf{Enc}(\mathsf{pk}, m)$, for some m) that a TREnc must fulfill, that trigger a pathological behavior of both the randomization and the tracing algorithms while honestly generated ciphertexts will never trigger that but with negligible probability. This behavior is defined in such a way that Rand is the identity function on that pathological ciphertexts. Moreover, computing such valid pathological ciphertexts is easy. That means that it is possible to satisfy the strong receipt-freeness of BeleniosRF while the answer of $\mathcal{O}\mathsf{receiptLR}$ can be to simply broadcast a ballot containing a pathological ciphertext without changing a single bit of it.

We turn to the description of the encryption scheme. The key generation is unchanged and we show how we slightly adapt the other algorithms of our TREnc below.

$\mathsf{Enc}'(\mathsf{PK}, m)$: to encrypt a message $m \in \mathbb{G}$, first run $(\mathsf{osk}, \mathsf{opk}) \leftarrow \mathsf{LinkGen}(\mathsf{PK})$ Then, conduct the following steps of $\mathsf{LEnc}'(\mathsf{PK}, \mathsf{lk}, m)$: first pick a random bit b so that $b = 1$ with negligible probability $2^{-\lambda}$. Then, run $\mathsf{LEnc}(\mathsf{PK}, \mathsf{lk}, m)$ with the slight modification in the computation of the tag in item 4, where we now compute the tag as $\tau' = H(\mathsf{opk}||b)$ instead of $\tau = H(\mathsf{opk})$. Output the ciphertext $\mathsf{CT}' = \mathsf{CT}||b$.

$\mathsf{Trace}'(\mathsf{PK}, \mathsf{CT}')$: Given $\mathsf{CT}' = \mathsf{CT}||b$, compute $\mathsf{opk} = \mathsf{Trace}(\mathsf{PK}, \mathsf{CT})$ and return $\tau' = H(\mathsf{opk}||b)$. *That is we redefine the trace as the new tag that depends on b.*

$\mathsf{Rand}'(\mathsf{PK}, \mathsf{CT}')$: Given $\mathsf{CT}' = \mathsf{CT}||b$, either return CT' with no modification if $b = 1$ or do the following. Compute $\tau' = \mathsf{Trace}'(\mathsf{PK}, \mathsf{CT}')$ and (honestly) run $\mathsf{Rand}(\mathsf{PK}, \mathsf{CT})$ except that the adaptation of the validity proof π in step 3 is made for the tag τ' (and not $\tau = H(\mathsf{opk})$). We thus get a randomized $\widetilde{\mathsf{CT}}$ from our TREnc and we return $\widetilde{\mathsf{CT}}' = \widetilde{\mathsf{CT}}||0$ in that case.

$\mathsf{Ver}'(\mathsf{PK}, \mathsf{CT}')$: Given $\mathsf{CT}' = \mathsf{CT}||b$, return $\mathsf{Ver}(\mathsf{PK}, \mathsf{CT})$ where the validity check in Equation 4 is made for the tag $\tau' = H(\mathsf{opk}||b)$ (and not $\tau = H(\mathsf{opk})$).

$\mathsf{Dec}'(\mathsf{SK}, \mathsf{PK}, \mathsf{CT}')$: If $\mathsf{Ver}(\mathsf{PK}, \mathsf{CT}') = 0$, output $\perp$. Otherwise, given $\mathsf{SK} = (\alpha, \beta)$ and $\mathbf{c} = (c_0, c_1, c_2)$ included in CT', compute and output $m = c_0 \cdot c_1^{-\alpha} \cdot c_2^{-\beta}$.

In the TCCA experiment, the adversary can send $(\mathsf{CT}', \mathsf{CT}')$ in the challenge phase, where $\mathsf{CT}' = \mathsf{CT}||1$. This pair will not be rejected as long as CT' is valid since the trace is trivially the same. We also note that a pair of ciphertext $\mathsf{CT}'_0, \mathsf{CT}'_1$ of the form $\mathsf{CT}'_0 = \mathsf{CT}_0||0$ and $\mathsf{CT}'_1 = \mathsf{CT}_1||1$, so with distinct bits, will not have the same trace except if we can compute a collision $H(\mathsf{opk}_0||0) = H(\mathsf{opk}_1||1)$. Moreover, the adversary cannot hope to flip the bit of the challenge ciphertext $(\mathsf{CT}^*||b)$ and receive its decryption by modifying the trace without contradicting the TBC-security of $(\mathbf{c}^*, \pi^*)$ for the tag $\tau^* = H(\mathsf{opk}^*, b)$. Thus, the TCCA security still holds and it is easy to see that the other relaxed notions are satisfied.

So, the BeleniosRF receipt-freeness definition will be satisfied for a voting scheme built from this encryption scheme, even though CT' appears, without any modification, on the public bulletin board, and the randomness used to compute it can then be used as a receipt. However this voting scheme won't satisfy the second condition of the definition of strong validity (Def. G.4).