Encryption Mechanisms for Receipt-Free and Perfectly Private Verifiable Elections

Thi Van Thao Doan[1(✉)], Olivier Pereira[1,2], and Thomas Peters[1]

[1] Université catholique de Louvain ICTEAM - Crypto Group, B-1348, Louvain-la-Neuve, Belgium
{thi.doan,olivier.pereira,thomas.peters}@uclouvain.be
[2] Microsoft Research, Redmond, WA, USA

Abstract. We design new encryption mechanisms that enable the design of the first universally verifiable voting schemes, supporting both receipt-freeness and everlasting privacy without assuming the existence of an anonymous channel.

Our schemes support the two most traditional election tallying methods: One is additively homomorphic, supporting elections in which votes simply need to be added, but decryption is only efficient for a message space of polylogarithmic size. The other is randomizable, is compatible with traditional mixnet-based tallying methods, and supports efficient message encoding, which makes it compatible with virtually any election type.

Our approach builds on the recently proposed traceable receipt-free encryption (TREnc) primitive to support the design of a perfectly private audit trail. In particular, we propose two TREnc that are secure under SXDH and rely on a public coin CRS (or on the random oracle model). This improves on previous TREnc mechanisms that required a structured CRS and is of independent interest. A prototype implementation of our mechanisms is proposed, which shows that ballot preparation and verification can be executed in less than a second.

Keywords: Traceable receipt-free encryption · Everlasting privacy · Perfectly private audit trail · Pairing-based cryptography

1 Introduction

Verifiable elections enable internal players and external observers to verify the validity of individual votes and the final election outcome, even in situations where potentially all participants have malicious intent. Verifiability is typically obtained through the use of a public bulletin board [1,13,14,18,34].

While being central to support public verifiability at scale, this bulletin board raises central issues in secret ballot elections. In order to guarantee the secrecy of the vote, a bulletin board will typically have ballots and/or voter names hidden by some form of encryption. This is a good solution to guarantee the computational privacy of the votes [6], but it does not address two other central concerns:

C. Pöpper and L. Batina (Eds.): ACNS 2024, LNCS 14583, pp. 257–287, 2024.
https://doi.org/10.1007/978-3-031-54770-6_11

1. The bulletin board may support vote selling or voter coercion. For instance, a voter who keeps track of all the random coins he used to prepare a ballot may be able to demonstrate how he voted to a third party, who would be able to recompute the encrypted ballot using the coins and claimed vote intent and confirm its presence on the bulletin board.
2. The bulletin board may raise long-term privacy concerns: when privacy is computational, encrypted votes (or encrypted voter identities) will eventually become public, either through cryptanalytic advances, or through the advent of new hardware, including quantum computers. This may have a chilling effect on voters who may not feel that they can vote freely.

As discussed in a recent review by Haines et al. [24], the design of secure and efficient voting protocols that offer both receipt-freeness (RF) and a perfectly private audit trail (PPAT) is a long-standing open problem. In particular, the few existing proposals that support these properties are either designed for in-person voting [32], or rely on the existence of an anonymous ballot submission channel [22,30], which seems hardly realistic in a large-scale election context (see further discussions in the related works section below).

An alternative approach to obtain receipt-freeness relies on the help of a ballot-box manager, who is trusted for receipt-freeness, but not for verifiability or for privacy, following a general approach pioneered by Hirt and Sako [25]. This approach led to the recent development of a new Traceable Receipt-Free Encryption (TREnc) primitive by Devillez et al. [20], which enables receipt-free ballot submission following the Hirt and Sako paradigm. In a nutshell, using a TREnc, voters can encrypt their vote intent, which will provide them with a trace and a ciphertext that can be submitted to the ballot-box manager. The trace can be kept by the voter for verifiability purpose and is actually independent of the vote itself, supporting receipt-freeness. The ballot-box manager then re-randomizes the TREnc ciphertext and posts the result on a public bulletin board. The security of the TREnc guarantees that the resulting ciphertext is distributed just like a fresh encryption of a vote with an identical trace (this is the strong randomization property) and that, if the trace did not change, then it must be the same vote that is still encrypted (this is the traceability). The voter can then verify on the bulletin board that a ciphertext with the correct trace is published, but is unable to explain the vote that is encrypted there to any third party. Finally, a TREnc guarantees that, even when ballots are computed with adversarially chosen randomness, no adversary can turn a re-randomized ballot into a related ballot that would have a different trace and contain a related vote (this is implied by the TCCA security). The existing TREnc mechanisms however do not support a PPAT and, as a side constraint, require a structured common reference string (CRS), which may be an obstacle in any practical deployment.

1.1 Our Contributions

We propose two new encryption mechanisms that make it possible to obtain both receipt-freeness (RF) and a perfectly private audit trail (PPAT) in a natural way, following the general structure of a single-pass voting system [7] for instance.

Our first encryption mechanism is additively homomorphic and suitable for elections with a homomorphic tally, that is, where the vote for each candidate can be encrypted as a 0 or a 1 counter, proven correct in zero-knowledge proof (ZK), and where the tally for each candidate is obtained by verifiably decrypting the homomorphic sum of all the ciphertexts provided for that candidate – this is the mechanism used by default in systems like Helios [2], Belenios [15] or ElectionGuard [4]. However, this encryption mechanism only supports a message space of polylogarithmic size, just as the exponential ElGamal encryption mechanism used in the previously cited systems, which makes it unsuitable to encrypt complex choice expressions in a single ciphertext.

Our second encryption mechanism addresses this limitation by supporting the encryption of arbitrary group elements and supporting efficient bijective mappings between bit strings and group elements, at the cost of losing the additive homomorphism. Still, this encryption mechanism is randomizable and compatible with traditional mixnets like Verificatum [36] that operate on arrays of group elements. Such mixnets have been used in national elections of various countries, including Norway, Estonia, and Switzerland.

From a technical point of view, our new encryption mechanisms are secure under symmetric external Diffie-Hellman assumption (SXDH) and provide new TREnc mechanisms that rely on a public-coin CRS. This is an advance over previous proposals, that is of independent interest, since the previously known mechanisms relied on a structured CRS, which may be complicated to generate in practice and introduces new trust assumptions. Here, a simple way of producing the CRS in practice would be to sample the outputs of a hash function modeled as a random oracle, which is already of common use in practical voting systems when implementing the Fiat-Shamir transform on sigma protocols.

Finally, we evaluate the efficiency of our new mechanisms. Our additively homomorphic mechanism produces ciphertexts in the two source groups of our pairing-friendly setting: they lie in $\mathbb{G}^{50} \times \hat{\mathbb{G}}^{46}$. Our mixnet-compatible mechanism produces slightly smaller ciphertexts in $\mathbb{G}^{47} \times \hat{\mathbb{G}}^{45}$. The gains essentially come from the inclusion, in the first case, of ZK proofs that a bit is encrypted, which is not needed for a mixnet-based tallying process. We also implemented our two mechanisms, relying on the MIRACL library for the group operations, and observed that both encryption operations require less than 0.3 s, and that the verification of the validity of a ciphertext takes less than a second.

1.2 Our Techniques

Commitment-Consistent Encryption. Our starting point for obtaining a PPAT is the use of a commitment-consistent encryption (CCE) scheme [19]. The CCE encryption of a message m provides two components: a perfectly hiding commitment com that comes out of a commitment scheme $(\mathsf{com}, \mathsf{open}) \leftarrow \mathsf{Com}(m)$, and an encryption enc of m and open that is provided together with a proof π_{cc} ensuring that $\mathsf{VerC}(\mathsf{com}, m, \mathsf{open}) = 1$, where VerC is the verification algorithm associated to Com. The proof π_{cc} is provided in order to guarantee that the CCE ciphertext is valid and that the tally will be computable. It can be augmented

with a proof π_{pub} that demonstrates that a valid opening of com (e.g., as a bit) is known.

When a CCE encryption scheme is used in a voting application, the value $D = (\mathsf{com}, \pi_{pub})$ is posted on a public bulletin board PB. If π_{pub} is perfect ZK, then this is perfectly hiding, as desired. Additionally, $CT = (\mathsf{enc}, \pi_{cc})$ is posted on a secret bulletin board SB, for use by the talliers. The election tally can then be computed using various techniques, as outlined in [19] for instance, preserving the PPAT. This approach however does not offer receipt-freeness, since the voter could use open as a receipt for his vote, for instance.

A TREnc on Top of CCE Encryption. In order to obtain the RF property, we then explore how to build a TREnc, whose ciphertexts contain a commitment-consistent encryption instead of an ElGamal encryption as in existing designs.

To satisfy all the TREnc properties that are needed need to achieve receipt-freeness in our voting scheme, all the components of $C = (D, CT)$ must be rerandomizable up to a trace which will allow voters to check the presence of their rerandomized commitments and proofs $D' = (\mathsf{com}', \pi'_{pub})$ on PB. Therefore, the traceability property must be supported by D and its randomization. For that purpose, we adapt the linearly homomorphic structure-preserving (LHSP) signature [28] techniques of [20] used during the computation of the encryption algorithm of their ad-hoc construction to our "commitment case". More precisely, the trace of a TREnc ciphertext C is the one-time LHSP verification key opk generated by the encryption algorithm. In a nutshell, the corresponding one-time LHSP secret key osk is used to sign a basis of a sub-vector space, where the vectors of this basis are derived from an internal CPA-encryption of the plaintext and its public key. The LHSP properties allow us to derive a signature on any rerandomization of this CPA part, and all its rerandomizations actually consist of the sub-vector space that is authenticated. Traceability comes from the fact that signing a CPA encryption of another plaintext requires authenticating a vector outside the linear spanned sub-space, which is unfeasible thanks to the unforgeability of the one-time LHSP signature scheme. Unfortunately, the unforgeability of the LHSP signatures cannot directly be used in our case to ensure that the rerandomized commitment com' still contains the same committed message. After all, there is just no meaning of which message is really contained in the perfectly hiding com' since it could be equally opened on *any* message. To restore this property and to contradict the security of the LHSP signatures when the committed message has been successfully modified, we use a dual-commitment compatible with $(\mathsf{Com}, \mathsf{VerC})$. To show traceability, we only have to turn the commitment public key into an extractable and perfectly binding mode at the start of the proof. We will thus have LHSP signatures and opk contained in D.

Finding the Right Tools. Finding most of the compatible building blocks is not straightforward, but only requires making careful choices and adapting techniques except for the TCCA property, and the simulation soundness in particular. Since we need rerandomizable proofs, we naturally focus on the SXDH-based

Groth-Sahai (GS) proof system that is also known to be malleable. Randomizing our TREnc ciphertexts C also requires adapting the statement under the GS proofs. Indeed, the witness underlying the commitment-consistent and validity proofs are subject to adaptation when com and Enc are rerandomized: the random coins are refreshed and the witness of the GS proofs may depend on these coins. In the perfect non-interactive witness indistinguishable (NIWI) setting to prove pairing-product equations, the common reference string (CRS) consists of random group elements of the two source groups and is thus public coin. However, NIWI proofs are not enough for our constructions as we need them to be zero-knowledge when we have to include an SXDH challenge in enc during the randomization in the challenge phase of the TCCA game, and for which the reduction is of course not given the random coin. There exist generic solutions to turn NIWI GS-proofs into ZK proofs but they are expensive. In spirit, we follow [29] which still partially relies on a generic OR-proof technique that makes it possible to prove another statement in the security proof than the one in the real execution of the scheme. The idea is that no adversary can use the second branch of the OR-proof for a TREnc ciphertext containing a different trace than the one of the challenge phase. In order to trigger the possibility of proving the second branch, we rely on the unforgeability of (yet another) one-time LHSP signature whose public key is generated in the key generation of the TREnc. Fortunately, this public key is a single uniformly distributed pair of group elements and is then public coin as well, while no one knows the corresponding signing key. The branch that is being proved is encoded as a vector of group elements with the following property: if the real statement is being proved, the vector only contains neutral elements (i.e., it is the null-vector, but we will use multiplicative notation); to simulate, we prove the second statement and the vector is non-trivial and lies in a one-dimensional subspace determined by the trace. Since, on the one hand, it is easy to compute a degenerated LHSP signature on the neutral vector from the public key, and, on the other hand, it is hard to compute an LHSP signature on a different one-dimensional subspace for another trace, simulation soundness holds for any proof with another trace than the one of the challenge. This vector as well as the (degenerated) LHSP signature are only given in a committed form with a GS-proof (with the same CRS) that the LHSP verification equation holds. Another difficulty in the TCCA proof is to switch the kind of encoded vector by simulating the randomization of one of the ciphertexts given in the challenge phase, but surprisingly this task can be handled only thanks to the perfect WI property of the GS-proof.

Following [20], we also need to commit-and-prove to the one-time LHSP signature generated during encryption (for the traceability) for technical reasons. Otherwise, even if it looks hard to embed subliminal information into the LHSP signatures related to com, we have no ground to prove the TCCA security. This additional layer of GS-proof solves the issue thanks to the perfect WI property. However, even when we should extract the witness of the proof of validity and consistency to figure out which branch the adversary tried to prove in a given ciphertext in a decryption query, this part of the proof related to the LHSP

signature for traceability must remain in the perfect WI mode. that is because we have to avoid leaking the internal bit of the game (i.e., which ciphertext between C_0 and C_1 have been rerandomized in the challenge phase) in an information-theoretic sense to conclude. To circumvent this opposing requirement, we use another GS CRS for the traceable part that can remain in the perfect WI mode.

1.3 Related Work

Receipt-Free Voting. The study of receipt-free elections was initiated by Benaloh and Tuinstra [5], who presented the first verifiable secret-ballot election protocols in which voters cannot prove to others how they voted. In order to achieve receipt-freeness, they required a physical voting booth to establish completely untappable channels between the voting authority and the voter. A year later, Sako and Kilian [35] argued that a one-way untappable channel is sufficient for this purpose. Additionally, they explained how to implement a receipt-free and universally verifiable voting system using the first verifiable mixnet. Thereafter, there has been a flurry of activity in the design and analysis of receipt-free voting protocols relying on the use of an untappable channel proposed by different authors. A prominent approach, that we outlined above and follow here, was proposed by Hirt and Sako [25], then simplified by Blazy et al. [9], refined by Chaidos et al. [12] and formalized by Devillez et al. [20]. Of course, many other approaches have been proposed in parallel and are out of scope of this work [21, 26, 34].

Voting with a PPAT. A recent and detailed account of the efforts towards voting with perfectly private ballots is provided by Haines et al. [24]. They identify the approach of commitment-consistent encryption, which we are using here, as one of the two strongest proposals, the other one being based on ballots that are secret-shared between a set of trustees [16]. We did not adopt the secret sharing approach here as it is more demanding to the voters, requiring a computational effort that grows linearly with the number of trustees that receive vote shares, and only offers privacy benefits over CCE if we assume that the voters have direct communication channels with every trustee, which may be quite demanding.

Voting with RF and PPAT. There are very few proposals that offer both RF and a PPAT, and they rely on the existence of anonymous channels for ballot submission, an assumption that we are avoiding here and that is hardly practical at a large scale.

The first is based on blind signatures [22, 33], where voters obtain a blindly signed voting token from an authority, which is then used to submit a ballot through an anonymous communication channel. Verifiability is hard to obtain in such a setting: a malicious authority can for instance produce tokens on behalf of abstaining voters and cast ballots in their stead.

A second approach was proposed by Locher and Haenni [30] and addresses the problem of eligibility verifiability by having voters registering a public key and submitting their ballot together with a proof that they know the secret

key matching one of these public keys, using a mechanism similar to list signatures [11]. Again, ballots are submitted using an anonymous communication channel.

Our work aims at developing solutions that offer a PPAT in the sense of Cuvelier et al. [19], together with receipt-freeness, following the definitional approach of Chaidos et al. [12].

1.4 Overview of Paper

We structure our paper as follows. Standard building blocks and the computational assumptions are introduced in Sect. 2. In Sect. 3, we present the intuition and the full description of our first construction to verifiably encrypt bits, followed by its theorem statements. The second construction is postponed to Appendix A due to space limit. Then, Sect. 4 shows the voting application implied by our combined primitive and describes an election based on homomorphic aggregation for the simple ballot case and an election with mixnet for the complex ballot case. To conclude, Sect. 5 makes some important remarks. The security analysis is deferred to Appendix B. This choice is compensated by the thorough overview of our techniques given above.

2 Background

We review some standard building blocks and introduce the corresponding notations.

2.1 Assumptions and Primitives

We will work in asymmetric bilinear groups and assume the existence of a bilinear-group generator $\mathcal{G}$ which takes the security parameter λ as input and outputs $\mathsf{pp} = (\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e, g, h, \hat{g}, p)$, where $\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T$ are groups of prime order $p > 2^{\mathsf{poly}(\lambda)}$, $g, h \xleftarrow{\$} \mathbb{G}$, $\hat{g} \xleftarrow{\$} \hat{\mathbb{G}}$ are random generators, and $e : \mathbb{G} \times \hat{\mathbb{G}}$ is a non-degenerate bilinear map. Our setting relies on the SXDH (symmetric external Diffie-Hellman) assumption, which states that the decisional Diffie-Hellman problem (DDH) [10] must be intractable in both $\mathbb{G}$ and $\hat{\mathbb{G}}$.

Assumption 1 (DDH). *Let λ be a security parameter and g be a generator of a group $\mathbb{G}$ of prime order $p > 2^{\mathsf{poly}(\lambda)}$. It is computationally hard to distinguish the tuple (g^a, g^b, g^{ab}) from the tuple (g^a, g^b, g^c) where $a, b, c \xleftarrow{\$} \mathbb{Z}_p$.*

Groth-Sahai Proofs. Groth-Shai (GS) proofs [23] offer an efficient approach to proving the satisfiability of quadratic equations in bilinear settings. On input pp, common reference strings (CRS) $\mathbf{u} \in \mathbb{G}^4$ and $\mathbf{v} \in \hat{\mathbb{G}}^4$ are generated to commit to groups elements of $\mathbb{G}$ and $\hat{\mathbb{G}}$. For instance, the commitments to $X \in \mathbb{G}$ and $\hat{Y} \in \hat{\mathbb{G}}$ are denoted by $\mathbf{C}_X$ and $\mathbf{C}_{\hat{Y}}$ respectively. In accordance with the GS standard

notation, we also define the linear maps: $\iota_1 : \mathbb{G} \to \mathbb{G}^2$ with $\iota_1 : X \mapsto (1, X)$ and $\iota_2 : \hat{\mathbb{G}} \to \hat{\mathbb{G}}^2$ with $\iota_2 : \hat{Y} \mapsto (1, \hat{Y})$.

Linearly Homomorphic Structure-Preserving Signatures (LHSP Signature). LHSP signature was introduced by Libert et al. [28] to perform linear computations on encrypted data. Structure-preserving property allows signing messages that are vectors of group elements, whereas the linearly homomorphic feature makes it possible to derive a signature on any linear combination of already signed vectors. In our context, we rely on a one-time LHSP signature scheme of [28] in the SXDH setting as in [27], where each voter signs only one linear subspace using his secret signing key.

$\mathsf{Gen}(\mathsf{pp}, \lambda, n)$: given the public parameter pp and the dimension $n \in \mathbb{N}$ of the subspace to be signed, pick $\chi_i, \gamma_i \xleftarrow{\$} \mathbb{Z}_p$ and compute $f_i = g^{\chi_i} h^{\gamma_i}$, for $i = 1$ to n. The private key is $\mathsf{sk} = \{(\chi_i, \gamma_i)\}_{i=1}^n$ and the public key is $\mathsf{pk} = \{f_i\}_{i=1}^n \in \mathbb{G}^n$.

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

$\mathsf{Ver}(\mathsf{pk}, \sigma, (M_1, \ldots, M_n))$: given a signature $\sigma = (\hat{Z}, \hat{R}) \in \hat{\mathbb{G}}^2$ and a vector $(M_1, \ldots, M_n)$, return 1 if and only if $(M_1, \ldots, M_n) \neq (1_{\hat{\mathbb{G}}}, \ldots, 1_{\hat{\mathbb{G}}})$ and $(\hat{Z}, \hat{R})$ satisfies

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

In our case, the LHSP signature has three key advantages. Firstly, it allows ciphertexts to be re-randomized and the adaptation of their signatures, while guaranteeing the non-malleability of the plaintext. Secondly, it notably ensures that it is infeasible to publicly compute a signature on a vector outside the linear span of originally signed vectors, which is essential for our system security. Thirdly, the verification Eq. (1), which is a pairing product equation, still holds for $(M_1, \ldots, M_n) = (1_{\hat{\mathbb{G}}}, \ldots, 1_{\hat{\mathbb{G}}})$ with a degenerated signature $(\hat{Z}, \hat{R}) = (1, 1)$. This feature allows hiding whether we trivially satisfy the equation or if we have a valid signature thanks to the Groth-Sahai proof system. We use it to implement an OR-technique useful to simulation soundness.

2.2 Traceable Receipt-Free Encryption (TREnc)

TREnc [20] is a public key encryption scheme (Gen, Enc, Dec), augmented with a 5-tuple of algorithms: LGen, on input a security parameter λ and a public encryption key PK, outputs a link key lk; LEnc encrypts a message m using (PK, lk) and outputs a ciphertext c. Trace outputs the trace t of c. Rand randomizes c to output a randomized ciphertext c'. Ver checks if a ciphertext is valid and outputs 1 if true, and 0 otherwise.

Verifiability. A TREnc is *verifiable* if no PPT adversary can produce, with non-negligible probability, a ciphertext that satisfies Ver but is not in the range

of Enc. In other words, Ver guarantees that a valid ciphertext is necessarily in the range of the honestly generated encryptions. More formally, for any efficient $\mathcal{A}$, $\Pr[c \notin \mathsf{Enc}(\mathsf{PK}, \cdot) \wedge \mathsf{Ver}(\mathsf{PK}, c) = 1 \mid (\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{Gen}(1^\lambda), c \leftarrow\!\$\ \mathcal{A}(\mathsf{PK}, \mathsf{SK})]$ is negligible. We denote the event of $\mathcal{A}$ winning this game as $\mathsf{Exp}^{\mathrm{ver}}_{\mathcal{A}}(\lambda) = 1$.

Strong Randomization. To achieve receipt-freeness, a ballot from a voter must be re-randomized before being placed on the PB. Strong randomization requires that the output of the Rand algorithm be indistinguishable from any encryption of the same message with the same link key. More precisely, 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: $\mathsf{Rand}(\mathsf{PK}, c) \approx_c \mathsf{LEnc}(\mathsf{PK}, \mathsf{lk}, m)$.

TCCA Security. Security against traceable chosen ciphertexts attacks, also called TCCA security, is a TREnc's central security requirement. An adversary $\mathcal{A}$, who has a public key and is allowed to access a decryption oracle, submits a pair of valid ciphertexts of its choice that have identical traces. One of the ciphertexts is randomized and returned to $\mathcal{A}$, who must decide which one it is. After receiving this challenge ciphertext, $\mathcal{A}$ can still query the decryption oracle, but only on ciphertexts that have a trace different from his challenge ciphertext. TREnc is said to be *TCCA secure* if no PPT adversary can decide which ciphertext was randomized. Achieving TCCA security implies a form of non-malleability of the trace of ciphertexts. This essentially guarantees the absence of a vote receipt and is formalized in the $\mathsf{Exp}^{\mathrm{tcca}}_{\mathcal{A}}(\lambda)$ game in Fig. 1.

Traceability. A TREnc is *traceable* if no efficient adversary $\mathcal{A}$ can produce another ciphertext that traces to the same trace and decrypts to a different message. The traceability property of the TREnc then guarantees that nobody (including the rerandomizing server and decryption-key holders) could have forged another valid ciphertext of another vote linked to the given ballot with non-negligible probability. This property is fundamental for the verifiability of an election and is defined in $\mathsf{Exp}^{\mathrm{trace}}_{\mathcal{A}}(\lambda)$ game of Fig. 1.

Link Traceability. TREnc allows the encryption of any message using a single link key and all resulting ciphertexts have the same trace. Thanks to this property, LEnc makes the TCCA game possible by encrypting different messages that trace to each other. This non-binding feature is essential for receipt-free voting.

Receipt-Freeness. To be receipt-free, TREnc relies on a semi-trusted entity called a ballot box manager. This entity checks the validity of the encrypted vote sent by the voter without requiring a secret key and then re-randomizes every valid ciphertext before posting it on the PB. Since the randomness contained in the published ballot is no longer under the control of the voter, he cannot prove how he voted. On the one hand, link traceability allows voters to vote for different messages with a single link key, preventing them from proving their vote. On the other hand, traceability ensures that no corrupted authority should be able to modify the encrypted vote while keeping the trace unchanged.

In a model where the voting client may be corrupted, strong randomization and TCCA security guarantees that the encryption hides the message. In

contrast, the traceability property plays an important role when the voting client is honest, and the re-randomization server might be corrupted.

Definition 2.1 (TREnc correctness). *A TREnc scheme is required to satisfy 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.*

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 message* m*, it holds that* $\mathsf{Ver}(\mathsf{PK}, \mathsf{Enc}(\mathsf{PK}, m)) = 1$.

$\mathsf{Exp}^{\mathrm{tcca}}_{\mathcal{A}}(\lambda)$

$(\mathsf{PK}, \mathsf{SK}) \leftarrow\!\$\ \mathsf{Gen}(1^\lambda)$
$(c_0, c_1, \mathsf{st}) \leftarrow\!\$\ \mathcal{A}_1^{\mathsf{Dec}(.)}(\mathsf{PK})$
$b \leftarrow\!\$\ \{0, 1\}$
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
$c^* \leftarrow\!\$\ \mathsf{Rand}(\mathsf{PK}, c_b)$
$b' \leftarrow\!\$\ \mathcal{A}_2^{\mathsf{Dec}^*(.)}(c^*, \mathsf{st})$
return $b' = b$

$\mathsf{Exp}^{\mathrm{trace}}_{\mathcal{A}}(\lambda)$

$(\mathsf{PK}, \mathsf{SK}) \leftarrow\!\$\ \mathsf{Gen}(1^\lambda)$
$(m, \mathsf{st}) \leftarrow\!\$\ \mathcal{A}_1(\mathsf{PK}, \mathsf{SK})$
$c \leftarrow\!\$\ \mathsf{Enc}(\mathsf{PK}, m)$
$c^* \leftarrow\!\$\ \mathcal{A}_2(c, \mathsf{st})$
if $\mathsf{Trace}(\mathsf{PK}, c) = \mathsf{Trace}(\mathsf{PK}, c^*)$ and
$\mathsf{Ver}(\mathsf{PK}, c^*) = 1$ and $\mathsf{Dec}(\mathsf{SK}, c^*) \neq m$
then return 1
else return 0

Fig. 1. The experiments of TCCA security, and traceability. In the TCCA game, $\mathcal{A}_2$ has access to a decryption oracle $\mathsf{Dec}^*(.)$ which, on input c, returns $\mathsf{Dec}(c)$ if $\mathsf{Trace}(\mathsf{PK}, c) \neq \mathsf{Trace}(\mathsf{PK}, c^*)$ and test otherwise.

2.3 Commitment Consistent Encryption (CCE)

CCE [19] is a cryptographic mechanism providing audit data for public verification that will never leak any information about the vote, even if the private keys are compromised or the cryptographic assumptions are broken.

To cast a ballot, voters are expected to encrypt their vote and produce a perfectly hiding commitment to the vote. The committed vote and an auxiliary value used to compute the commitment are called the openings for that commitment. The encryption is computed so that from any encrypted vote, it is possible to extract a commitment and an encryption of openings for that commitment. To verify the validity of the ballots, voters also have to provide a non-interactive

zero-knowledge proof demonstrating the consistency between these components. The commitment is then cast on PB, whereas the encryption of the openings and the consistency proof are sent to SB. Since the election audit data in PB is perfectly hiding, we can ensure the confidentiality of the votes.

However, it is easy to observe that if a voter is willing to sell his vote, he can store the openings of the commitment and communicate it to an adversary. Thus, although CCE makes an e-voting system everlastingly private, it is not designed to protect against the vote-selling/buying threat. In other words, such a CCE protocol is not receipt-free.

3 The Construction of Our Scheme

Sections 3.1 and 3.2 describe our first construction of a commitment-consistent TREnc tailored to simple ballot. That is, the message space is bits encoded as scalars (in the exponents), and ciphertexts contain publicly verifiable proof of so. Our second construction tailored for complex ballot is deferred to Appendix A, where the message is a group element. In Sect. 3.3, we give the correctness and the security theorem statements of the construction. Finally, we provide a performance evaluation of our encryption algorithm in Sects. 3.4.

3.1 Description

Gen(1^λ)**:** Choose bilinear groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ of prime order $p > 2^{\mathsf{poly}(\lambda)}$, pick $g, g_1, h_1 \xleftarrow{\$} \mathbb{G}$ and $\hat{g}, \hat{h}, \hat{g}_1, \hat{h}_1, \hat{g}_2, \hat{h}_2 \xleftarrow{\$} \hat{\mathbb{G}}$.

1. Pick random $\{(\alpha_i, \beta_i)\}_{i=1}^3 \xleftarrow{\$} \mathbb{Z}_p$ and set $\{f_i\}_{i=1}^3 = g_1^{\alpha_i} h_1^{\beta_i}$.
2. To commit to groups elements of $\mathbb{G}$ and $\hat{\mathbb{G}}$ respectively, we generate one Groth-Sahai CRS $\mathbf{u} = (\vec{u}_1, \vec{u}_2)$ in $\mathbb{G}^4$ and two others $\mathbf{v} = (\vec{v}_1, \vec{v}_2)$ and $\mathbf{v}' = (\vec{v}'_1, \vec{v}'_2)$ in $\hat{\mathbb{G}}^4$ such that $\vec{u}_1 = (u_{11}, u_{12})$, $\vec{u}_2 = (u_{21}, u_{22})$, $\vec{v}_1 = (v_{11}, v_{12})$, $\vec{v}_2 = (v_{21}, v_{22})$, $\vec{v}'_1 = (v'_{11}, v'_{12})$, and $\vec{v}'_2 = (v'_{21}, v'_{22})$ are generated in the perfect NIWI mode.
3. Pick random $\hat{f}_1, \hat{f}_2 \leftarrow \hat{\mathbb{G}}$ that will be used as a verification key for the LHSP signature but for which no one knows the corresponding secret key.

The private and public keys respectively are $\mathsf{SK} = \{(\alpha_i, \beta_i)\}_{i=1}^3$ and $\mathsf{PK} = (g, g_1, h_1, \hat{g}, \hat{h}, \hat{g}_1, \hat{h}_1, \hat{g}_2, \hat{h}_2, \{f_i\}_{i=1}^3, \hat{f}_1, \hat{f}_2, \mathbf{u}, \mathbf{v}, \mathbf{v}')$.

Enc(PK, m)**:** To encrypt $m \in \mathbb{Z}_p$, first run $\mathsf{LGen}(\mathsf{PK})$: Generate a key pair $(\mathsf{osk}, \mathsf{opk})$ for the one-time linearly homomorphic signature from the public generators g_1, h_1 to sign vectors of dimension 3. Let the signing key $\mathsf{lk} = \mathsf{osk} = \{(\eta_i, \zeta_i)\}_{i=1}^3$, the corresponding public key is $\mathsf{opk} = \{k_i\}_{i=1}^3 = \{g_1^{\eta_i} h_1^{\zeta_i}\}_{i=1}^3$. Then, conduct the following steps of $\mathsf{LEnc}(\mathsf{PK}, \mathsf{lk}, m)$:

1. In the ciphertext CT:
   (a) Compute $M = g^m \in \mathbb{G}$. For random $r, q \xleftarrow{\$} \mathbb{Z}_p$, compute the commitments $\hat{d}_1 = \hat{g}^m \hat{g}_1^r \hat{h}_1^q \in \hat{\mathbb{G}}$, $\hat{d}_2 = \hat{g}_2^r \hat{h}_2^q \in \hat{\mathbb{G}}$ and the openings $R = g^r \in \mathbb{G}, Q = g^q \in \mathbb{G}$. Choose $\theta \xleftarrow{\$} \mathbb{Z}_p$, compute the ciphertexts of M, R, and Q respectively as $\mathbf{c}_m = (c_0, c_1, c_2) = (Mf_1^\theta, g_1^\theta, h_1^\theta)$, $c_r = Rf_2^\theta$, and $c_q = Qf_3^\theta$.
   (b) Commit to the openings using the Groth-Sahai CRS by computing $\mathbf{C}_M = \iota_1(M)\vec{u}_1^{z_1}\vec{u}_2^{z_2} \in \mathbb{G}^2$, $\mathbf{C}_R = \iota_1(R)\vec{u}_1^{r_1}\vec{u}_2^{r_2} \in \mathbb{G}^2$, and $\mathbf{C}_Q = \iota_1(Q)\vec{u}_1^{t_1}\vec{u}_2^{t_2} \in \mathbb{G}^2$ for random $z_1, z_2, r_1, r_2, t_1, t_2 \xleftarrow{\$} \mathbb{Z}_p$, then derive the commitment $\mathbf{C}_{f_1} = \iota(c_0)/\mathbf{C}_M$, $\mathbf{C}_{f_2} = \iota(c_r)/\mathbf{C}_R$, and $\mathbf{C}_{f_3} = \iota(c_q)/\mathbf{C}_Q$.
   (c) To allow simulating the proof, set the bit $\bar{b} = 1$ and compute $G = g^{\bar{b}} \in \mathbb{G}$ and $\hat{H} = \hat{h}^{\bar{b}} \in \hat{\mathbb{G}}$. Commit to G, $\hat{H}$, and $\hat{H}^\theta$ respectively to have $\mathbf{C}_G = \iota_1(G)\vec{u_1}^{w_1}\vec{u_2}^{w_2} \in \mathbb{G}^2$, $\mathbf{C}_{\hat{H}} = \iota_2(\hat{H})\vec{v_1}^{x_1}\vec{v_2}^{x_2} \in \hat{\mathbb{G}}$, and $\mathbf{C}_\theta = \iota_2(\hat{H}^\theta)\vec{v_1}^{x_3}\vec{v_2}^{x_4} \in \hat{\mathbb{G}}$ for $w_1, w_2, x_1, x_2, x_3, x_4 \xleftarrow{\$} \mathbb{Z}_p$. To make sure G and $\hat{H}$ are well-formed, compute GS proof π_b such that

$$e(g, \boxed{\hat{H}}) = e(\boxed{G}, \hat{h}) \tag{2}$$

   For the sake of simplicity, we signify that the group element represented in the box is the one that is committed in the corresponding commitment. For example, in Eq. 2, $\hat{H}$ and G are committed in $\mathbf{C}_{\hat{H}}$ and $\mathbf{C}_G$ respectively.
   (d) To make sure CT is well-formed, compute the GS proof π_θ to ensure that $(c_1, c_2, c_0/M, c_r/R, c_q/Q)$ are in the form of $(g_1, h_1, f_1, f_2, f_3)^\theta$. In other words, these equations below must be satisfied with $\boxed{\hat{H}^\theta}$, $\boxed{f_1^\theta}$, $\boxed{f_2^\theta}$, and $\boxed{f_3^\theta}$ respectively being committed in $\mathbf{C}_\theta, \mathbf{C}_{f_1}, \mathbf{C}_{f_2}$, and $\mathbf{C}_{f_3}$.

$$\begin{aligned}
e(c_1, \boxed{\hat{H}}) &= e(g_1, \boxed{\hat{H}^\theta}) && \text{(a)}\\
e(c_2, \boxed{\hat{H}}) &= e(h_1, \boxed{\hat{H}^\theta}) && \text{(b)}\\
e(\boxed{f_1^\theta}, \boxed{\hat{H}}) &= e(f_1, \boxed{\hat{H}^\theta}) && \text{(c)}\\
e(\boxed{f_2^\theta}, \boxed{\hat{H}}) &= e(f_2, \boxed{\hat{H}^\theta}) && \text{(d)}\\
e(\boxed{f_3^\theta}, \boxed{\hat{H}}) &= e(f_3, \boxed{\hat{H}^\theta}) && \text{(e)}
\end{aligned} \tag{3}$$

   (e) Return $CT = (\mathbf{c}_m, c_r, c_q, \mathbf{C}_{\hat{H}}, \mathbf{C}_\theta, \pi_b, \pi_\theta) \in \mathbb{G}^{25} \times \hat{\mathbb{G}}^{20}$.
2. In the commitment D:
   (a) For the proof of the openings for commitments:
      – The GS proof of openings π_{open} needs to make sure that the values committed in $\mathbf{C}_M, \mathbf{C}_R, \mathbf{C}_Q, \mathbf{C}_G$ in CT are the openings of the commitments $\hat{d}_1, \hat{d}_2$ in D. To put it differently, π_{open} must satisfy that

$$\begin{aligned}
e(\boxed{M}, \hat{g}) \cdot e(\boxed{R}, \hat{g}_1) \cdot e(\boxed{Q}, \hat{h}_1) &= e(\boxed{G}, \hat{d}_1) && \text{(a)}\\
e(\boxed{R}, \hat{g}_2) \cdot e(\boxed{Q}, \hat{h}_2) &= e(\boxed{G}, \hat{d}_2) && \text{(b)}
\end{aligned} \tag{4}$$

where $\boxed{M}$, $\boxed{R}$, $\boxed{Q}$ are values committed in $\mathbf{C}_M, \mathbf{C}_R, \mathbf{C}_Q$ in their respective order.

Thus, the proofs π_θ computed in the CT and π_{open} in D constitute the *proof of consistency* between $\hat{d}_1, \hat{d}_2$ and $\mathbf{c}_m, c_r, c_q$, i.e., the commitments can be opened by encrypted values in the ciphertexts.

(b) For traceability property:

– Sign each row of the matrix T using $\mathsf{lk} = \mathsf{osk}$, resulting in signatures $\hat{\sigma}_1, \hat{\sigma}_2, \hat{\sigma}_3$, where $\hat{\sigma}_i = (\hat{Z}_i, \hat{R}_i) \in \hat{\mathbb{G}}^2$ for $i = 1, 2, 3$.

$$T = \begin{pmatrix} \hat{g} & \hat{d}_1 & \hat{d}_2 \\ 1 & \hat{g}_1 & \hat{g}_2 \\ 1 & \hat{h}_1 & \hat{h}_2 \end{pmatrix} \tag{5}$$

– To allow strong randomizability, commit to $\hat{\sigma}_1$ using the GS CRS $\mathbf{v}'$ by computing $C_{\hat{Z}} = \iota_2(\hat{Z}_1)\vec{v}_1'^{l_1}\vec{v}_2'^{l_2}$ and $C_{\hat{R}} = \iota_2(\hat{R}_1)\vec{v}_1'^{l_3}\vec{v}_2'^{l_4}$ for random scalars $l_1, l_2, l_3, l_4 \xleftarrow{\$} \mathbb{Z}_p$.

– To ensure that $\hat{\sigma}_1$ is a valid one-time LHSP signature on $(\hat{g}, \hat{d}_1, \hat{d}_2)$, compute the proof π_{sig} such that

$$e(g_1, \boxed{\hat{Z}_1}) \cdot e(h_1, \boxed{\hat{R}_1}) = e(k_1, \hat{g}) \cdot e(k_2, \hat{d}_1) \cdot e(k_3, \hat{d}_2) \tag{6}$$

where $\boxed{\hat{Z}_1}$ and $\boxed{\hat{R}_1}$ are committed in $C_{\hat{Z}}, C_{\hat{R}}$ respectively.

(c) For TCCA security:

– Set $(A, B) = (1_{\mathbb{G}}, 1_{\mathbb{G}})$ as a degenerated LHSP signature, and $X = g/G = g^{1-\bar{b}} \in \mathbb{G}$. Since $\bar{b} = 1$, $X = 1_{\mathbb{G}}$. The commitment of X is computed by $\mathbf{C}_X = \iota_1(g)/\mathbf{C}_G \in \mathbb{G}^2$. Commit to A and B to have $\mathbf{C}_A = \iota_1(A)\vec{u}_1^{a_1}\vec{u}_2^{a_2}$ and $\mathbf{C}_B = \iota_1(B)\vec{u}_1^{b_1}\vec{u}_2^{b_2}$ for $a_1, a_2, b_1, b_2 \xleftarrow{\$} \mathbb{Z}_p$.

– The randomizable simulation-sound proof π_{ss} must ensure that

$$e(\boxed{A}, \hat{g}) \cdot e(\boxed{B}, \hat{h}) = e(g/\boxed{G}, \hat{f}_1\hat{f}_2^{\tau}) \tag{7}$$

where $\tau = \mathsf{Hash}(\mathsf{opk})$. In the honest case, 7 is trivially fulfilled. In the simulated case, $\bar{b} \neq 1$ and (A, B) must be a valid LHSP signature on $(X, X^{\tau}) \neq (1, 1)$ with verification keys being public elements $(\hat{f}_1, \hat{f}_2)$.

(d) For well-formedness proof of a vote:

The vote m must be 0 or 1. To this end, we commit to $\hat{M} = \hat{g}^m$ to have $\mathbf{C}_{\hat{M}} = \iota_2(\hat{M})\vec{v}_1^{s_1}\vec{v}_2^{s_2}$ for random scalars $s_1, s_2 \xleftarrow{\$} \mathbb{Z}_p$. The proof π_{01} is computed such that

$$\begin{aligned} &e(\boxed{M}, \hat{g}) = e(g, \boxed{\hat{M}}) && \text{(a)} \\ &e(\boxed{M}, \hat{g}/\boxed{\hat{M}}) = 1 && \text{(b)} \end{aligned} \tag{8}$$

(e) Return the commitment part $D = (\hat{d}_1, \hat{d}_2, \mathbf{C}_{\hat{M}}, \mathbf{C}_M, \mathbf{C}_R, \mathbf{C}_Q, \mathbf{C}_G, \mathbf{C}_{\hat{Z}}, \mathbf{C}_{\hat{R}}, \mathbf{C}_A, \mathbf{C}_B, \pi_{open}, \hat{\sigma}_2, \hat{\sigma}_3, \pi_{sig}, \pi_{ss}, \pi_{01}, \mathsf{opk}) \in \mathbb{G}^{25} \times \hat{\mathbb{G}}^{26}$.

At the end of the encryption, output $C = (CT, D) \in \mathbb{G}^{50} \times \hat{\mathbb{G}}^{46}$.

Trace(PK, C**):** Parse PK and C as above, and output opk in the obvious way.
Rand(PK, C**):** If PK and $C = (CT, D)$ do not parse as the outputs of Gen and Enc, abort. Otherwise, conduct the steps as follows:

1. Randomizing CT:
   (a) Parse the CPA encryption part $\mathbf{c}_m, c_r, c_q$, pick $\theta', r', q' \xleftarrow{\$} \mathbb{Z}_p$, set $R' = g^{r'}, Q' = g^{q'}$, and compute $\mathbf{c}'_m = (c'_0, c'_1, c'_2) = \mathbf{c}_m \cdot (f_1, g_1, h_1)^{\theta'} = (M f_1^{\theta+\theta'}, g_1^{\theta+\theta'}, h_1^{\theta+\theta'}), c'_r = c_r \cdot R' f_2^{\theta'}$, and $c'_q = c_q \cdot Q' f_3^{\theta'}$.
   (b) Adapt the commitments $\mathbf{C}'_G = \mathbf{C}_G \cdot \vec{u_1}^{w'_1} \vec{u_2}^{w'_2}$, and $\mathbf{C}'_{\hat{H}} = \mathbf{C}_{\hat{H}} \cdot \vec{v_1}^{x'_1} \vec{v_2}^{x'_2}$ for $w'_1, w'_2, x'_1, x'_2 \xleftarrow{\$} \mathbb{Z}_p$. Likewise, randomize the commitments $C'_{\hat{H}} = \mathbf{C}_{\hat{H}} \cdot \vec{v_1}^{x'_1} \vec{v_2}^{x'_2}$, $\mathbf{C}'_\theta = \mathbf{C}_\theta \cdot \iota_2(1) \cdot \iota_2(\hat{H}^{\theta'}) \vec{v_1}^{x'_3} \vec{v_2}^{x'_4}$, $\mathbf{C}'_M = \mathbf{C}_M \cdot \vec{u_1}^{z'_1} \vec{u_2}^{z'_2}$, $\mathbf{C}'_R = \mathbf{C}_R \cdot \iota_1(R') \vec{u_1}^{r'_1} \vec{u_2}^{r'_2}$, $\mathbf{C}'_Q = \mathbf{C}_Q \cdot \iota_1(Q') \vec{u_1}^{t'_1} \vec{u_2}^{t'_2}$ for $x'_1, x'_2, x'_3, x'_4, z'_1, z'_2, r'_1, r'_2, t'_1, t'_2 \xleftarrow{\$} \mathbb{Z}_p$. The derived commitments are then $\mathbf{C}'_{f_1} = \iota(c'_0)/\mathbf{C}'_M$, $\mathbf{C}'_{f_2} = \iota(c'_r)/\mathbf{C}'_R$; $\mathbf{C}'_{f_3} = \iota(c'_q)/\mathbf{C}'_Q$.
   (c) Adapt the proof π'_θ and π'_b accordingly.
   (d) Return $CT' = (\mathbf{c}'_m, c'_r, c'_q, \mathbf{C}'_{\hat{H}}, \mathbf{C}'_\theta, \pi'_b, \pi'_\theta)$.
2. Randomizing D:
   (a) For proof of openings π_{open}:
      i. Randomize the commitments $\hat{d}'_1 = \hat{d}_1 \cdot \hat{g}_1^{r'} \hat{h}_1^{q'} = \hat{g}^m \hat{g}_1^{r+r'} \hat{h}_1^{q+q'}$, $\hat{d}'_2 = \hat{d}_2 \cdot \hat{g}_2^{r'} \hat{h}_2^{q'} = \hat{g}_2^{r+r'} \hat{h}_2^{q+q'}$ for the same r', q' in CT.
      ii. Update the corresponding proof π'_{open}.
   (b) For the proof of signature π_{sig}:
      i. Implicitly adapt the committed signature $\hat{\sigma}_1$ of the tracing part by computing $\tilde{\sigma}_1 = (\tilde{Z}_1, \tilde{R}_1) = (\hat{Z}_2^{r'} \hat{Z}_3^{q'}, \hat{R}_2^{r'} \hat{R}_3^{q'})$ which consists of a one-time LHSP signature on $(1, \hat{g}_1, \hat{g}_2)^{r'} \cdot (1, \hat{h}_1, \hat{h}_2)^{q'}$ for opk.
      ii. Adapt the commitment $\mathbf{C}'_{\hat{Z}} = \mathbf{C}_{\hat{Z}} \cdot \iota_2(\tilde{Z}_1) \vec{v_1}'^{l'_1} \vec{v_2}'^{l'_2}$ and $\mathbf{C}'_{\hat{R}} = \mathbf{C}_{\hat{R}} \cdot \iota_2(\tilde{R}_1) \vec{v_1}'^{l'_3} \vec{v_2}'^{l'_4}$ for some random $l'_1, l'_2, l'_3, l'_4 \xleftarrow{\$} \mathbb{Z}_p$, which should commit to the valid one-time LHSP signature $\hat{\sigma}'_1 = \hat{\sigma}_1 \hat{\sigma}_2^{r'} \hat{\sigma}_3^{q'}$ on $(g, \hat{d}'_1, \hat{d}'_2)$ for opk. Then, randomize the proof π'_{sig}.
   (c) For the proof of simulation soundness π_{ss}:
      i. Adapt the commitment $\mathbf{C}_X$ corresponding to $\mathbf{C}'_G$ by computing $\mathbf{C}'_X = \iota_1(g)/\mathbf{C}'_G$. Similarly, computing $\mathbf{C}'_A = \mathbf{C}_A \cdot \vec{u_1}^{a'_1} \vec{u_2}^{a'_2}$ and $\mathbf{C}'_B = \mathbf{C}_B \cdot \vec{u_1}^{b'_1} \vec{u_2}^{b'_2}$ for some $a'_1, a'_2, b'_1, b'_2 \xleftarrow{\$} \mathbb{Z}_p$.
      ii. Adapt the proof π_{ss} to have π'_{ss}.
   (d) For well-formedness proof of a vote:
   Adapt the commitment $\mathbf{C}'_{\hat{M}} = \mathbf{C}_{\hat{M}} \cdot \vec{v_1}^{s'_1} \vec{v_2}^{s'_2}$ for $s'_1, s'_2 \xleftarrow{\$} \mathbb{Z}_p$. Similarly, adapt the proof π_{01} to have π'_{01}.
   (e) Return $D' = (\hat{d}'_1, \hat{d}'_2, \mathbf{C}'_{\hat{M}}, \mathbf{C}'_M, \mathbf{C}'_R, \mathbf{C}'_Q, \mathbf{C}'_G, \mathbf{C}'_A, \mathbf{C}'_B, \mathbf{C}'_{\hat{Z}}, \mathbf{C}'_{\hat{R}}, \hat{\sigma}_2, \hat{\sigma}_3, \pi'_{sig}, \pi'_{open}, \pi'_{ss}, \pi'_{01}, \mathsf{opk})$.

At the end of the randomization, output $C = (CT', D')$.

Ver (PK, C)**:** Abort and output 0 if either PK or C fails to parse correctly. Else, check the validity of the LHSP signatures $\hat{\sigma}_2, \hat{\sigma}_3$ respectively on $(1, \hat{g}_1, \hat{g}_2)$ and $(1, \hat{h}_1, \hat{h}_2)$ with respect to opk, as well as all the Groth-Sahai proofs with $\tau = \mathsf{Hash}(\mathsf{opk})$ and output 0 if at least one of them fails; otherwise, output 1. (All the verification equations are given in Sect. 3.2.)

Dec(SK, C)**:** If $\mathrm{Ver}(\mathsf{PK}, C) = 0$, output $\perp$. Otherwise, given $\mathsf{SK} = \{(\alpha_i, \beta_i)\}_{i=1}^{3}$ and $(\mathbf{c}_m = (c_0, c_1, c_2), c_r, c_q)$ included in CT, output $M = c_0 \cdot c_1^{-\alpha_1} \cdot c_2^{-\beta_1}$, $R = c_r \cdot c_1^{-\alpha_2} \cdot c_2^{-\beta_2}$, and $Q = c_q \cdot c_1^{-\alpha_3} \cdot c_2^{-\beta_3}$.

3.2 Verification Equations

We now turn to the specification of the verification equations of the Groth-Sahai proofs that must be satisfied by valid ciphertexts produced using this first construction. While they are not necessary to follow the security proofs, we expand them here in order to have a clear view about the cost of publicly verifying ciphertexts, which will be evaluated through a prototype implementation below.

Ver (PK, C)**:** Abort and output 0 if either PK or C fails to parse correctly. Then, privately verify the first verification, which concerns the CPA encryption part, while the remaining four will be checked publicly on PB.

1. The CPA encryption part is well-formed, i.e., $(c_1, c_2, c_0/M, c_r/R, c_q/Q)$ are all in the form of the same exponent; and $(G, \hat{H})$ are also raised to the same exponent. To hold, the proofs π_θ, π_b in CT (of Eq. 3, Eq. 2) and commitments $\mathbf{C}_{f_1}, \mathbf{C}_{f_2}, \mathbf{C}_{f_3}, \mathbf{C}_{\hat{H}}, \mathbf{C}_\theta, \mathbf{C}_G$ must satisfy:

$$E(c_1, \mathbf{C}_{\hat{H}}) = E(g_1, \mathbf{C}_\theta) \cdot E(\pi_{\theta,a}[0], \vec{v}_1) \cdot E(\pi_{\theta,a}[1], \vec{v}_2)$$
$$E(c_2, \mathbf{C}_{\hat{H}}) = E(h_1, \mathbf{C}_\theta) \cdot E(\pi_{\theta,b}[0], \vec{v}_1) \cdot E(\pi_{\theta,b}[1], \vec{v}_2)$$

and

$$\begin{aligned} E(\mathbf{C}_{f_i}, \mathbf{C}_{\hat{H}}) =& E(\iota_1(f_i), \mathbf{C}_\theta) \cdot E(\vec{u}_1, \pi_{\theta,j}[0]) \cdot E(\vec{u}_2, \pi_{\theta,j}[1]) \\ & \cdot E(\pi_{\theta,j}[2], \vec{v}_1) \cdot E(\pi_{\theta,j}[3], \vec{v}_2) \end{aligned}$$

for $(i, j) \in \{(1, c), (2, d), (3, e)\}$ and

$$\begin{aligned} E(\iota_1(g), \mathbf{C}_{\hat{H}}) =& E(\mathbf{C}_G, \iota_2(\hat{h})) \cdot E(\vec{u}_1, \pi_b[0]) \cdot E(\vec{u}_2, \pi_b[1]) \cdot \\ & E(\pi_b[2], \vec{v}_1) \cdot E(\pi_b[3], \vec{v}_2) \end{aligned}$$

2. The values committed in $\mathbf{C}_M, \mathbf{C}_R, \mathbf{C}_Q, \mathbf{C}_G$ are the openings of the commitments in Eq. 4. That means

$$E(\mathbf{C}_M, \hat{g}) \cdot E(\mathbf{C}_R, \hat{g}_1) \cdot E(\mathbf{C}_Q, \hat{h}_1) = E(\mathbf{C}_G, \hat{d}_1) \cdot E(\mathbf{u}, \pi_{open,a})$$
$$E(\mathbf{C}_R, \hat{g}_2) \cdot E(\mathbf{C}_Q, \hat{h}_2) = E(\mathbf{C}_G, \hat{d}_2) \cdot E(\mathbf{u}, \pi_{open,b})$$

where $E(\mathbf{u}, \pi_{open,i}) = E(\vec{u}_1, \pi_{open,i}[0]) \cdot E(\vec{u}_2, \pi_{open,i}[1])$ with $i \in \{a, b\}$. The verifications 1. and 2. constitute a consistency between $\hat{d}_1, \hat{d}_2$ in D and $\mathbf{c}_m, c_r, c_q$ in CT, i.e., the commitments can be opened by encrypted values in the ciphertexts.

3. The committed signature of the tracing part is valid, i.e., $\hat{\sigma}_1 = (\hat{Z}_1, \hat{R}_1)$ is a valid one-time LHSP signature on the vector $(\hat{g}, \hat{d}_1, \hat{d}_2)$. To this end, the commitments $\mathbf{C}_{\hat{Z}}, C_{\hat{R}}$ and the proof π_{sig} must ensure that

$$\begin{aligned} E(g_1, \mathbf{C}_{\hat{Z}}) \cdot E(h_1, \mathbf{C}_{\hat{R}}) =& E(k_1, \iota_2(\hat{g})) \cdot E(k_2, \iota_2(\hat{d}_1)) \cdot E(k_3, \iota_2(\hat{d}_2)) \cdot \\ & E(\pi_{sig}[0], \vec{v}_1') \cdot E(\pi_{sig}[1], \vec{v}_2') \end{aligned}$$

4. The committed values of the simulation part are valid, i.e., (A, B) must be a valid LHSP signature on (X, X^τ) by verifying

$$E(\mathbf{C}_A, \hat{g}) \cdot E(\mathbf{C}_B, \hat{h}) = E(\iota_1(g)/\mathbf{C}_G, \hat{f}_1 \hat{f}_2^\tau) \cdot E(\vec{u}_1, \pi_{ss}[0]) \cdot E(\vec{u}_2, \pi_{ss}[1])$$

5. The vote m is 0 or 1 using the proof π_{01} from Eq. 8

$$\begin{aligned} E(\mathbf{C}_M, \iota_2(\hat{g})) =& E(\iota_1(g), \mathbf{C}_{\hat{M}}) \cdot E(\vec{u}_1, \pi_{01,a}[0]) \cdot E(\vec{u}_2, \pi_{01,a}[1]) \cdot \\ & E(\pi_{01,a}[2], \vec{v}_1) \cdot E(\pi_{01,a}[3], \vec{v}_2) \\ E(\mathbf{C}_M, \iota_2(\hat{g})/\mathbf{C}_{\hat{M}}) =& E(\vec{u}_1, \pi_{01,b}[0]) \cdot E(\vec{u}_2, \pi_{01,b}[1]) \cdot \\ & E(\pi_{01,b}[2], \vec{v}_1) \cdot E(\pi_{01,b}[3], \vec{v}_2) \end{aligned}$$

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

3.3 Security Analysis

The above scheme enjoys (perfect) correctness. Moreover, its security solely relies on the SXDH assumption as claimed below. All the proofs are given in Appendix B.

Theorem 3.1. *The above scheme is perfectly strongly randomizable.*

Theorem 3.2. *The above scheme is TCCA-secure under the SXDH assumption and the collision resistance of the hash function. We have the advantage* $|\Pr[\mathsf{Exp}^{\mathrm{tcca}}_{\mathcal{A}}(\lambda) = 1] - \frac{1}{2}| \leq \epsilon_{cr} + 6\epsilon_{sxdh} + \frac{4}{p}$.

Theorem 3.3. *The above scheme is traceable under the SXDH assumption. More precisely, we have* $\Pr[\mathsf{Exp}^{\mathrm{trace}}_{\mathcal{A}}(\lambda) = 1] \leq 5\epsilon_{sxdh} + \frac{1}{p}$.

Theorem 3.4. *The above scheme is verifiable under the SXDH assumption. More precisely, for any adversary* $\mathcal{A}$*, we have* $\Pr[\mathsf{Exp}^{\mathrm{ver}}_{\mathcal{A}}(\lambda) = 1] \leq 3\epsilon_{sxdh} + \frac{1}{p}$.

3.4 Efficiency

Up to constant factors, the encryption scheme we just described and the one we describe in Appendix A are optimal in the sense of Cramer, Gennaro and Schoenmakers [17]: the ballot size and the voter computational load do not depend on the number of voters nor on the number of authorities, the computational workload of the tallying authorities grows linearly with the number of voters and candidates.

In order to evaluate the constants, we built a C implementation of the ballot preparation (key generation, encryption) and verification algorithms using the MIRACL Core Cryptographic Library [31]. The implementation, which can be found on https://github.com/uclcrypto/TREnc-PPAT, is carried out on an average commodity laptop equipped with an Intel i5-1245U processor running Ubuntu 22.04. The time unit is seconds and all our results are averaged over 100 runs. The running time of verification includes all verification equations in Sect. 3.1 for both individual and universal verification. Likewise, the encryption process timing also includes signature and proof computation. To provide at least 128-bit security, we use a BN curve [3] on a 462-bit prime field, so-called BN462. As seen in Table 1, it appears that the cost of computing ballots for

Table 1. Time for key generation, encryption, and verification of one ballot.

Tally type	Gen	Enc	Ver
Homomorphic	0.023	0.228	0.802
Mixnet	0.019	0.214	0.782

both instances is almost similar and largely under a second. However, there is a slight difference in the verification timing of the two methods. This is because a mixnet-based tally does not require a well-formedness proof of the vote, whereas a homomorphic tally does. We note that the computation of multiple ciphertexts could also largely benefit of fixed-base exponentiation methods: these costs can then grow much more slowly than linearly with the number of ciphertexts to be computed.

4 Application to E-Voting

One important application of our scheme is the construction of single-pass voting systems [8], where voters interact with the system only by submitting their ballots. The described protocol involves four entities as introduced in TREnc, consisting of: *voters*, who have the right to vote; *election administrator (EA)*, who is in charge of setting up the election and generating PK and SK. A *ballot box manager* is responsible for randomizing the ballots of the voters. A *tallier* is in charge of correctly tallying the ballot box and providing the correctness proof of the tally. Also, it provides tallying results on a public view PB for verifiability.

Our proposed voting protocol is defined as a tuple of probabilistic polynomial-time algorithms based on the two most crucial tallying techniques: homomorphic aggregation, tailored for elections with a small number of candidates, and mixnet that is suitable for elections with complex ballots.

$\mathsf{SetUp}(1^\lambda)$: On input security parameter 1^λ, generates the public and secret keys $(\mathsf{PK}, \mathsf{SK})$ of the election.

Vote(id, v): Upon receiving a voter id and a vote v, outputs a ballot $\mathsf{b} = (CT, D)$.
Valid(b): On input ballot b, outputs 0 or 1. The algorithm outputs 1 if and only if the ballot satisfies all verification equations.
ProcessBallot(b): On the input ballot b, outputs an updated ballot b', a re-randomization of b where $\mathsf{b}' = (CT', D')$.
TraceBallot(b): On input a commitment D, outputs a trace t. The trace is the information that a voter can use to track his ballot, using VerifyBallot.
Append(PB, SB, b) : On input PB, SB, and ballot b, appends D to PB and CT to SBif Valid(b) = 1.
VerifyBallot(PB, t): On input the public board PB and a trace t, outputs 0 or 1. This algorithm is used by voters to check if their ballot has been processed and recorded properly.
Tally(PB, SB, SK): On input PB, SB, and private key SK, outputs the tally result and a proof of correctness. Depending on the tallying technique, runs HomoTally or MixnetTally correspondingly.
VerifyResult(PB): On input PB, result of the tally and proof of the tally, outputs 0 or 1. Depending on the tallying technique, runs either HomoVerify or MixnetVerify.

It is implicit that PK is given to all these algorithms except SetUp.

Following [20], we describe our voting scheme based on a TREnc with the difference that only the perfectly private parts of our ballots are published on PB. More precisely, EAs first generate the election public and secret keys with SetUp by running Gen of our TREnc. The public key PK is published and stored on the PB, and shares of SK are only known by the tallier (SK can be securely generated in a distributed way in our prime-order groups using standard techniques). Each voter can then prepare a ballot b and submit it to the ballot box manager using the Vote algorithm that runs the encryption of the vote using our TREnc. The validity of the ballot is defined as the validity of our D and CT output by Vote(id, v). Although the ballot will be randomized, a voter can store TraceBallot(b) that is defined as the trace of the TREnc ciphertext and confirm if it has been correctly recorded on PB by utilizing VerifyBallot(PB, t). After receiving a ballot, the ballot box manager checks its validity and that no ballot with the same trace was recorded before. Invalid ballots are dropped and valid ones will go through Append(PB, SB, b) after being re-randomized by ProcessBallot(b) thanks to Rand. As said, Append(PB, SB, b) simply computes $\mathsf{PB} \leftarrow \mathsf{PB}||D$ and $\mathsf{SB} \leftarrow \mathsf{PB}||CT$ from (previously rerandomized) $\mathsf{b} = (CT, D)$. Once every voter has cast a vote, the tallier checks the validity of each ballot using Valid(b). A tallying protocol is then carried out based on the ballot type and the election outcome is published. To verify the election result, anyone can utilize VerifyResult(PB) by referring to the content of PB, and which can be based on common techniques.

4.1 Voting Scheme with a Homomorphic Tally

One of the two main approaches for tallying an election is homomorphic aggregation. The homomorphic property makes it possible to homomorphically combine

a number of ballots to compute the encrypted sum of the votes. Then only the sum is decrypted instead of individual votes so that the secrecy of an individual's ballot is preserved. Since the vote can be only 0 or 1, it can be encoded as a group element and subsequently decoded by an exhaustive search of the plaintext. Thus, the voter needs to add a non-interactive randomizable proof of vote well-formedness to the public commitment part (see Eq. 8). The tallier computes tally by $\mathsf{HomoTally}$ algorithm as follows.

1. *Aggregation*: For all l valid CT in SB, the tallier performs element-wise multiplication of the encryption $(\mathbf{c}_m, c_r, c_q)$, obtaining a result vector $\mathbf{v} = (\prod_i \mathbf{c}_m, \prod_i c_r, \prod_i c_q)$ for $i = 1, \ldots l$.
2. *Decryption*: The tallier decrypts $\mathbf{v}$ in order to obtain the openings $(M, R, Q) = \mathsf{Dec}(\mathsf{SK}, \mathbf{v})$, then finds m and appends (m, R, Q) to PB.

Since both the commitment and the encryption schemes are additively homomorphic, the votes are homomorphically combined into a single ciphertext containing the final result, which is then decrypted by the tallier. To check that the tally matches the posted votes, anyone can run $\mathsf{HomoVerify}$ algorithm as follows.

1. Multiply all the commitments $(\hat{d}_1, \hat{d}_2)$ element-wise for l valid entries on PB to obtain a commitment on the election outcome $(\mathsf{com}_1, \mathsf{com}_2)$.
2. Verify that (m, R, Q) provided by the tallier are openings of the outcome commitment by checking if the given equations are satisfied

$$e(g^m, \hat{g}) \cdot e(R, \hat{g}_1) \cdot e(Q, \hat{h}_1) = e(g, \mathsf{com}_1)$$
$$e(R, \hat{g}_2) \cdot e(Q, \hat{h}_2) = e(g, \mathsf{com}_2)$$

Given that the commitment scheme is binding, it makes sure that the only openings that the authorities are able to provide come from an honest tallying process. Moreover, its perfectly hiding property can guarantee the perfect ballot privacy of the whole audit trail.

4.2 Voting Scheme with a Mixnet Tally

Unlike homomorphic tallying, verifiable mixnet-based systems decrypt individual ballots after anonymization, which disassociates encrypted ballots from their corresponding voters. This anonymization procedure will be performed by shuffling the votes through several shuffling centers (so-called mixers). Since each shuffled ballot is decrypted individually, its validity is verified by the fact that the decrypted vote and auxiliary values are the openings of the corresponding commitment. As a result, the voter is not required to compute the well-formedness proof of the vote. Due to page limitations, our adapted encryption scheme of mixnet tallying is presented in the Appendix A, which is not additively homomorphic anymore, but still randomizable. Thus, there are no specific concerns regarding the necessary randomization properties for mixing. We sketch the $\mathsf{MixnetTally}$ algorithm in the following.

1. *Stripping*: On each input of $C = (CT, D)$, the authorities only keep the encryption $(\mathbf{c}_m, \mathbf{c}_r, c_s)$ in CT and the commitments (d_1, d_2) in D, obtaining an encryption vector $\mathbf{v} = \{(\mathbf{c}_m^i, \mathbf{c}_r^i, c_s^i)\}_{i=1}^l$ and corresponding commitment vector $\mathbf{d} = \{(d_1^i, d_2^i)\}_{i=1}^l$ of l valid ballots.
2. *Permutation Selection*: A random permutation π is chosen and a validity proof P_π for π is computed.
3. *Shuffle*: The mixers shuffle $(\mathbf{v}, \mathbf{d})$, resulting in $(\mathbf{v}', \mathbf{d}')$. While $\mathbf{v}'$ is kept private on SB, $\mathbf{d}'$ is posted on PB. Additionally, two commitment consistent proofs are provided with respect to the permutation π: $P_\mathbf{v}$ shows that $\mathbf{v}'$ is a shuffle on $\mathbf{v}$ and $P_\mathbf{d}$ shows that $\mathbf{d}'$ is a shuffle on $\mathbf{d}$. $P_\mathbf{v}$ and $P_\mathbf{d}$ are then posted on SB and PB respectively.
4. *Decryption*: After verifying the proofs, the encryption in $\mathbf{v}'$ is decrypted to have the message M and auxiliary values $\hat{R}$ and $\hat{S}$. The results are published on PB.

Since the published proofs do not disclose the permutation used in the mixing process or the decryption key, it would not violate any anonymity. Thus, everyone can verify if the election outcome is the correct decryption of the shuffled valid votes using the $\mathsf{MixnetVerify}$ algorithm below.

1. *Verification of the permutation*: One can verify the proof P_π of the chosen permutation π and abort if it fails.
2. *Verification of the proof of shuffle*: One can verify the validity of the proof $P_\mathbf{d}$ and abort if it fails.
3. *Verification of the openings*: One can verify if decrypted values of $\mathbf{v}'$ provided on PB are valid openings for the shuffled commitments in $\mathbf{d}'$ and abort otherwise.

5 Conclusion

Our paper proposes two encryption mechanisms for verifiable elections that supports both receipt-freeness and a perfectly private audit trail. To the best of our knowledge, this is the first proposal that can achieve these properties without relying on the presence of an anonymous channel for submitting the ballots.

On our way, we develop new traceable receipt-free encryption (TREnc) mechanisms that are secure under SXDH, assuming a public coin CRS. This last assumption brings a noticeable benefit over the existing mechanisms, which required a structured CRS, bringing the question of the practical generation of this CRS, and of the underlying trust assumptions.

We demonstrate the efficiency of our mechanism through a prototype implementation. While demanding, they still support encryption and ciphertext verification under a second of time. It would be appealing to explore solutions that could reduce the complexity of this encryption process, both in time and in space.

Acknowledgments. Thomas Peters is a research associate of the Belgian Fund for Scientific Research (F.R.S.-FNRS). This work has been funded in part by the Walloon Region through the project CyberExcellence (convention number 2110186).

A Scheme Description for Complex Ballots

Gen(1^λ): Choose bilinear groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ of prime order $p > 2^{\mathsf{poly}(\lambda)}$ together with $g, h, g_1, h_1 \xleftarrow{\$} \mathbb{G}$ and $\hat{g}, \hat{h} \xleftarrow{\$} \hat{\mathbb{G}}$.

1. Pick random $\{(\alpha_i, \beta_i)\}_{i=1}^2 \xleftarrow{\$} \mathbb{Z}_p$ and set $\{\hat{f}_i\}_{i=1}^2 = \hat{g}^{\alpha_i}\hat{h}^{\beta_i}$. Pick random $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_p$ and set $f = g^\alpha h^\beta$.
2. Generate Groth-Sahai CRS $\mathbf{u} = (\vec{u}_1, \vec{u}_2) \in \mathbb{G}^4$, $\mathbf{u}' = (\vec{u}'_1, \vec{u}'_2) \in \mathbb{G}^4$ and $\mathbf{v} = (\vec{v}_1, \vec{v}_2) \in \hat{\mathbb{G}}^4$ to commit to groups elements of $\mathbb{G}$ and $\hat{\mathbb{G}}$, where $\vec{u}'_1 = (u'_{11}, u'_{12}) = (g, h)$, $\vec{u}'_2 = (u'_{21}, u'_{22}) = (g_1, h_1)$, $\vec{v}_1 = (v_{11}, v_{12})$, and $\vec{v}_2 = (v_{21}, v_{22})$ are generated in the perfect NIWI mode.
3. Pick random $k_1, k_2 \leftarrow \mathbb{G}$ that will be used as a verification key for the LHSP signature.

The private key is $\mathsf{SK} = (\alpha_1, \beta_1, \alpha_2, \beta_2, \alpha, \beta)$ and the public key $\mathsf{PK} = (g, h, g_1, h_1, \hat{g}, \hat{h}, f, \hat{f}_1, \hat{f}_2, k_1, k_2, \mathbf{u}, \mathbf{v}, \mathbf{u}')$.

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

1. In the ciphertext CT:
   (a) For random $r, s \xleftarrow{\$} \mathbb{Z}_p$, compute the commitments $d_1 = Mg^r h^s \in \mathbb{G}, d_2 = g_1^r h_1^s \in \mathbb{G}$ and the openings $\hat{R} = \hat{g}^r \in \hat{\mathbb{G}}, \hat{S} = \hat{g}^s \in \hat{\mathbb{G}}$. Randomly choose $\theta, \gamma \xleftarrow{\$} \mathbb{Z}_p$, compute the ciphertexts of M, $\hat{R}$, and $\hat{S}$ respectively as $\mathbf{c}_m = (c_m^0, c_m^1, c_m^2) = (Mf^\theta, g^\theta, h^\theta)$, $\mathbf{c}_r = (c_r^0, c_r^1, c_r^2) = (\hat{R}\hat{f}_1^\gamma, \hat{g}^\gamma, \hat{h}^\gamma)$, and $c_s = \hat{S}\hat{f}_2^\gamma$.
   (b) Commit to the openings using the Groth-Sahai CRS by computing $\mathbf{C}_M = \iota_1(M)\vec{u}_1^{z_1}\vec{u}_2^{z_2}$, $\mathbf{C}_{\hat{R}} = \iota_1(\hat{R})\vec{v}_1^{r_1}\vec{v}_2^{r_2}$, and $\mathbf{C}_{\hat{S}} = \iota_1(\hat{S})\vec{v}_1^{t_1}\vec{v}_2^{t_2}$ for random $z_1, z_2, r_1, r_2, t_1, t_2 \xleftarrow{\$} \mathbb{Z}_p$. For the sake of simplicity, from now we denote the GS commitments as $\mathbf{C}_M = \mathsf{Com}(\mathbf{u}, M), \mathbf{C}_{\hat{R}} = \mathsf{Com}(\mathbf{v}, \hat{R})$, and $\mathbf{C}_{\hat{S}} = \mathsf{Com}(\mathbf{v}, \hat{S})$. Next, derive the commitments $\mathbf{C}_f = \iota_1(c_m^0)/\mathbf{C}_M$, $\mathbf{C}_{\hat{f}_1} = \iota_2(c_r^0)/\mathbf{C}_{\hat{R}}$, and $\mathbf{C}_{\hat{f}_2} = \iota_2(c_s)/\mathbf{C}_{\hat{S}}$.
   (c) To allow simulating the proof, set the bit $\bar{b} = 1$ and compute $G = g^{\bar{b}} \in \mathbb{G}$ and $\hat{G} = \hat{g}^{\bar{b}} \in \hat{\mathbb{G}}$. Commit to G, $\hat{G}$ to have $\mathbf{C}_G = \mathsf{Com}(\mathbf{u}, G)$, $\mathbf{C}_{\hat{G}} = \mathsf{Com}(\mathbf{v}, \hat{G})$. Compute GS proof π_b such that $e(g, \hat{G}) = e(G, \hat{g})$.
   (d) To ensure CT is well-formed, the proof π_θ is computed to make sure that $(c_m^1, c_m^2, c_m^0/M)$ and $(c_r^1, c_r^2, c_r^0/\hat{R}, c_s/\hat{S})$ are in the form of $(g, h, f)^\theta$ and $(\hat{g}, \hat{h}, \hat{f}_1, \hat{f}_2)^\gamma$ respectively. To do that, commit also to $\hat{G}^\theta$ and G^γ such that $\mathbf{C}_\theta = \mathsf{Com}(\mathbf{v}, \hat{G}^\theta)$ and $\mathbf{C}_\gamma = \mathsf{Com}(\mathbf{u}, G^\gamma)$, and compute a GS proof π_θ that

$$e(c_m^1, \hat{G}) = e(g, \hat{G}^\theta), e(c_m^2, \hat{G}) = e(h, \hat{G}^\theta), \quad e(G, c_r^1) = e(G^\gamma, \hat{g}),$$
$$e(G, c_r^2) = e(G^\gamma, \hat{h}), e(f^\theta, \hat{G}) = e(f, \hat{G}^\theta), \quad e(G, \hat{f}_1^\gamma) = e(G^\gamma, \hat{f}_1),$$
$$e(G, \hat{f}_2^\gamma) = e(G^\gamma, \hat{f}_2).$$

(e) Return $CT = (\mathbf{c}_m, \mathbf{c}_r, c_s, \mathbf{C}_G, \mathbf{C}_\theta, \mathbf{C}_\gamma, \pi_b, \pi_\theta) \in \mathbb{G}^{27} \times \hat{\mathbb{G}}^{26}$.

2. In the commitment D:

(a) *The proof of the openings for commitments*: The proof of openings π_{open} needs to make sure that the values committed in $\mathbf{C}_M, \mathbf{C}_{\hat{R}}, \mathbf{C}_{\hat{S}}$, $\mathbf{C}_{\hat{G}}$ in CT are the openings of the commitments. In other words, $\mathbf{C}_M, \mathbf{C}_{\hat{R}}, \mathbf{C}_{\hat{S}}$, and $\mathbf{C}_{\hat{G}}$ must ensure that $e(M, \hat{g}) \cdot e(g, \hat{R}) \cdot e(h, \hat{S}) = e(d_1, \hat{G})$ and $e(g_1, \hat{R}) \cdot e(h_1, \hat{S}) = e(d_2, \hat{G})$.

(b) *Traceability property*: Sign each row of the matrix T using $\mathsf{lk} = \mathsf{osk}$ to have signatures $\sigma_1, \sigma_2, \sigma_3$, where $\sigma_i = (Z_i, R_i) \in \mathbb{G}^2$ for $i = 1, 2, 3$.

$$T = \begin{pmatrix} g & d_1 & d_2 \\ 1 & g & g_1 \\ 1 & h & h_1 \end{pmatrix}$$

Next, commit to σ_1 using $\mathbf{u}'$ with $\mathbf{C}_Z = \mathsf{Com}(\mathbf{u}', Z_1)$ and $\mathbf{C}_R = \mathsf{Com}(\mathbf{u}', R_1)$. To ensure that σ_1 is a valid one-time LHSP signature on (g, d_1, d_2), compute the proof $\pi_{sig} \in \hat{\mathbb{G}}^2$ such that $e(Z_1, \hat{g}) \cdot e(R_1, \hat{h}) = e(g, \hat{y}_1) \cdot e(d_1, \hat{y}_2) \cdot e(d_2, \hat{y}_3)$.

(c) *TCCA security*: Set $\hat{A} = 1_{\hat{\mathbb{G}}}, \hat{B} = 1_{\hat{\mathbb{G}}}, \hat{X} = \hat{g}/\hat{G} = \hat{g}^{1-\bar{b}}$, and $\tau = \mathsf{Hash}(\mathsf{opk})$. Commit to $\hat{A}$ and $\hat{B}$ using CRS $\mathbf{v}$. Compute the proof π_{ss} that $e(g, \hat{A}) \cdot e(h, \hat{B}) = e(k_1 k_2^\tau, \hat{g}/\hat{G})$.

(d) Return $D = (d_1, d_2, \mathbf{C}_M, \mathbf{C}_{\hat{R}}, \mathbf{C}_{\hat{S}}, \mathbf{C}_{\hat{G}}, \mathbf{C}_Z, \mathbf{C}_R, \mathbf{C}_{\hat{A}}, \mathbf{C}_{\hat{B}}, \pi_{open}, \sigma_2, \sigma_3, \pi_{sig}, \pi_{ss}, \mathsf{opk}) \in \mathbb{G}^{20} \times \hat{\mathbb{G}}^{19}$.

At the end of the encryption, output $C = (CT, D) \in \mathbb{G}^{47} \times \hat{\mathbb{G}}^{45}$.

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

Rand(PK, C): If PK and $C = (CT, D)$ do not parse as the outputs of Gen and Enc, abort. Otherwise, conduct the similar steps as presented in $\mathsf{Rand}(\mathsf{PK}, C)$ (Sect. 3.1). At the end of the randomization, output the ciphertext $C' = (CT', D')$.

Ver (PK, C): First, abort and output 0 if either PK or C fails to parse correctly. Second, verify the validity of the signatures σ_2 and σ_3 on the 2 last rows of the matrix T, and output 0 if it does not hold. Third, verify all the provided GS proofs $\pi_b, \pi_\theta, \pi_{open}, \pi_{sig}$, and π_{ss} regarding their the corresponding equations. The first two proofs will be privately verified, which concerns the CPA encryption part, while the others will be checked publicly on PB. If at least one of these checks fails, output 0; otherwise, output 1.

Dec(SK, C): If $\mathrm{Ver}(\mathsf{PK}, C) = 0$, output $\perp$. Otherwise, given $\mathsf{SK} = (\alpha_1, \beta_1, \alpha_2, \beta_2, \alpha, \beta)$ and $(\mathbf{c}_m = (c_m^0, c_m^1, c_m^2), \mathbf{c}_r = (c_r^0, c_r^1, c_r^2), c_s)$ included in CT, output $M = c_m^0 \cdot {c_m^1}^{-\alpha} \cdot {c_m^2}^{-\beta}$, $\hat{R} = c_r^0 \cdot {c_r^1}^{1-\alpha_1} \cdot {c_r^2}^{-\beta_1}$, and $\hat{S} = c_s \cdot {c_r^1}^{1-\alpha_2} \cdot {c_r^2}^{-\beta_2}$.

The security analysis of this second scheme directly follows that of our first construction.

B Deferred Proofs

B.1 Correctness

The construction satisfies TREnc's correctness as defined in Definition 2.1.

Correctness of encryption compatibility By construction, we define Enc such that the distributions of $\mathsf{Enc}(\mathsf{PK}, m)$ and $\mathsf{LEnc}(\mathsf{PK}, \mathsf{LGen}(\mathsf{PK}), m)$ are identical.

Correctness of link traceability For every PK in the range of Gen, the scheme runs $\mathsf{LGen}(\mathsf{PK})$ to output a key pair $(\mathsf{osk}, \mathsf{opk})$ for the one-time linearly homomorphic signature, where $\mathsf{opk} = f(\mathsf{osk})$ for a deterministic function f. Then, for every $\mathsf{lk} = \mathsf{osk}$ in the range of $\mathsf{LGen}(\mathsf{PK})$, $\mathsf{LEnc}(\mathsf{PK}, \mathsf{lk}, m)$ produces a ciphertext C, where $\mathsf{Trace}(\mathsf{PK}, C) = f(\mathsf{osk}) = \mathsf{opk}$. That is, $\mathsf{Trace}(\mathsf{PK}, \mathsf{LEnc}(\mathsf{PK}, \mathsf{lk}, \cdot))$ is the constant function $f(\mathsf{osk}) = \mathsf{opk}$.

Correctness of publicly traceable randomization As described in 3.1, the trace opk is kept unchanged in randomization step. Thus, we have $\mathsf{Trace}(\mathsf{PK}, C) = \mathsf{Trace}(\mathsf{PK}, \mathsf{Rand}(\mathsf{PK}, C))$ by definition. Additionally, in Rand algorithm, we honestly randomize the CPA part of the ciphertext, where $\mathbf{c}'_m = \mathbf{c}_m \cdot (f_1, g_1, h_1)^{\theta'} = (M f_1^{\theta+\theta'}, g_1^{\theta+\theta'}, h_1^{\theta+\theta'})$ with $\theta' \xleftarrow{\$} \mathbb{Z}_p$. Obviously, $\mathbf{c}'_m$ is distributed exactly as a fresh CPA encryption of m since $\theta + \theta'$ is random over $\mathbb{Z}_p$. There exists no random θ' that can modify the message, even the coin might not have been taken from a uniform distribution. Hence, $\mathsf{Dec}(\mathsf{SK}, C) = \mathsf{Dec}(\mathsf{SK}, \mathsf{Rand}(\mathsf{PK}, C))$.

Correctness of honest verifiability Given a ciphertext C in an honest range of $\mathsf{Enc}(\mathsf{PK}, m)$, there exists random coins that explain how to compute the ciphertext. This always leads to valid GS proofs and valid LHSP signatures. Based on that, we have verifiability since all the verification equations are satisfied. In other words, thanks to the perfect correctness of GS proofs and LHSP signatures, if C is honestly generated, for all the coins, we have validity or $\mathsf{Ver}(\mathsf{PK}, \mathsf{Enc}(\mathsf{PK}, m)) = 1$.

B.2 Strong Randomizability

Theorem 3.1. *The TREnc is perfectly 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 distributions $\{\mathsf{Rand}(\mathsf{PK}, c)\}$ and $\{\mathsf{LEnc}(\mathsf{PK}, \mathsf{lk}, m)\}$ are identical.*

Proof. Given a ciphertext $C = (CT, D)$ in the range of $\mathsf{Enc}(\mathsf{PK}, m)$, for some message m and internal link key $\mathsf{lk} = \mathsf{osk}$, the perfect correctness of honest verifiability of our TREnc implies that C is valid. It is easy to see that the opening values R, Q are fully redistributed as uniform group elements during rerandomization. The CPA part is then also fully rerandomized and distributed as a fresh CPA part. In the WI mode, valid GS-proofs can also be perfectly rerandomized

and fully redistributed after adaptation. Finally, the LHSP signatures on the last two rows of the T-matrix are deterministic. The indistinguishability is actually perfect.

B.3 TCCA Security

Theorem 3.2. *The above scheme is TCCA-secure under the SXDH assumption and the collision resistance of the hash function. More precisely, we have* $\left|\Pr[\mathsf{Exp}^{\mathrm{tcca}}_{\mathcal{A}}(\lambda)=1]-\frac{1}{2}\right| \leq \epsilon_{cr}+6\epsilon_{sxdh}+\frac{4}{p}$.

Proof. We consider a sequence of games. In Game i, we denote by S_i the event that an adversary $\mathcal{A}$ wins by correctly guessing the internal random bit b of the game, which makes the game output 1.

$\mathsf{Game}_1(\lambda)$**:** This is the real game as described in the experiment Fig. 1. By definition, $\Pr[S_1]=\Pr[\mathsf{Exp}^{\mathrm{tcca}}_{\mathcal{A}}(\lambda)=1]$.

$\mathsf{Game}_2(\lambda)$**:** In this game, we introduce a failure event F_2 which causes this game to abort and output a random bit if the adversary produces two valid ciphertexts C and C' as output of Enc such that $\mathsf{Hash}(\mathsf{opk})=\mathsf{Hash}(\mathsf{opk}')$ but $\mathsf{opk}\neq\mathsf{opk}'$. This even prevents the situation when $\mathcal{A}$ can successfully use the same tag with different signatures in decryption queries after the challenge phase. F_2 implies a collision on the hash function, so $\Pr[F_2]=\epsilon_{cr}$. We thus have, $|\Pr[S_2]-\Pr[S_1]|\leq\Pr[F_2]=\epsilon_{cr}$.

$\mathsf{Game}_3(\lambda)$**:** This game is as Game 2 except that we introduce a failure event which occurs during the challenge phase if $\mathcal{A}$ can produce the valid ciphertexts C_0, C_1 but $(\hat{\sigma}_2^{(0)},\hat{\sigma}_3^{(0)})\neq(\hat{\sigma}_2^{(1)},\hat{\sigma}_3^{(1)})$. The event should be aborted since the challenge ciphertext C^* has the same values of $(\hat{\sigma}_2^*,\hat{\sigma}_3^*)$ as the ones in C_0 or C_1. This causes a distinguishability between them. Obviously, $|\Pr[S_0]-\Pr[S_1]|$ is bounded by the probability that $(\hat{\sigma}_2^{(0)},\hat{\sigma}_3^{(0)})$ and $(\hat{\sigma}_2^{(1)},\hat{\sigma}_3^{(1)})$ are 2 distinct signatures on the same vector. Thus, $|\Pr[S_3]-\Pr[S_2]|\leq\epsilon_{sxdh}$.

$\mathsf{Game}_4(\lambda)$**:** This game is the same as Game 3 except in the way we generate the challenge ciphertext C^* from C_b in the randomization step. When we generate PK, we compute $\hat{f}_1,\hat{f}_2$ in such a way that they are corresponding verification keys for a signing key $\mathsf{sk}_{\mathsf{lhsp}}$ of a one-time linearly homomorphic signature in order to sign vectors of dimension $n=2$, given the common public parameters $\hat{g},\hat{h}$. We keep in memory $\mathsf{sk}_{\mathsf{lhsp}}$ and output $\mathsf{pk}_{\mathsf{lhsp}}=\{\hat{f}_1,\hat{f}_2\}$. Since the distribution of the output is not changed, it is indistinguishable from $\mathcal{A}$'s view. The simulated randomization is as follows:

1. Randomizing D^*
   (a) For the proof of openings π^*_{open}
   - Randomize the commitments $\hat{d}_1^*=\hat{d}_1^{(b)}\cdot\hat{g}_1^{r^*}\hat{h}_1^{q^*}$, $\hat{d}_2^*=\hat{d}_2^{(b)}\cdot\hat{g}_2^{r^*}\hat{h}_2^{q^*}$ for $r^*,q^*\xleftarrow{\$}\mathbb{Z}_p$.
   - Switch $\bar{b}=0$, then we have $G^*=g^{\bar{b}}=1_{\mathbb{G}}\in\mathbb{G}$. Re-compute the commitment of G^* by $\mathbf{C}^*_G=\mathsf{Com}(\mathbf{u},1_{\mathbb{G}})$. Since CRS $\mathbf{u}$ 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, it is not able to distinguish between $\mathbf{C}_G = \mathsf{Com}(\mathbf{u}, g)$ and $\mathbf{C}^*_G = \mathsf{Com}(\mathbf{u}, 1_{\mathbb{G}})$.
- Similarly, update randomized commitments $\mathbf{C}^*_M = \mathsf{Com}(\mathbf{u}, 1_{\mathbb{G}})$, $\mathbf{C}^*_R = \mathsf{Com}(\mathbf{u}, 1_{\mathbb{G}})$, and $\mathbf{C}^*_Q = \mathsf{Com}(\mathbf{u}, 1_{\mathbb{G}})$. The Eqs. 4.a and 4.b are verified as valid since both sides of the equations are equal to 1. Then, update the simulated proof $\pi^*_{open} = (\pi^*_{4.a}, \pi^*_{4.b})$ with corresponding randomness. Since GS proof is WI, the simulated proof cannot be distinguished from a real one.

(b) For the proof of signature π^*_{sig}, it is done as usual as Rand would do.

(c) For the proof of simulation soundness π^*_{ss}
- Since $\bar{b}$ is switched to 0, $X^* = g^{1-\bar{b}} = g \in \mathbb{G}$. Adapt the commitment of X^* to be $\mathbf{C}^*_X = \iota_1(g)/\mathbf{C}^*_G = \mathsf{Com}(\mathbf{u}, g)$.
- We simulate the proof π^*_{ss} by resigning the message $(X^*, X^{*\tau^*})$ from scratch using the secret key $\mathsf{sk}_{\mathsf{lhsp}}$ at step 2c. That is, when we randomize $\mathbf{C}^{(b)}_A, \mathbf{C}^{(b)}_B$ from an adversary, first computing the LHSP signature (A^*, B^*) on $(X^*, X^{*\tau^*})$. In other words, $(A^*, B^*) = \mathsf{Sign}(\mathsf{sk}_{\mathsf{lhsp}}, (X^*, X^{*\tau^*}))$, where $\tau^* = \mathsf{Hash}(\mathsf{opk}_0) = \mathsf{Hash}(\mathsf{opk}_1) = \tau^b$. The Eq. 7 is still valid since $\hat{f}_1, \hat{f}_2$ was generated as the public verification keys corresponding to $\mathsf{sk}_{\mathsf{lhsp}}$. Indeed, since $\mathbf{C}'_A, \mathbf{C}'_B$ computed in Rand and $\mathbf{C}^*_A, \mathbf{C}^*_B$ are indistinguishable under NIWI CRS, their distributions are exactly the same in the adversary's view.
- Commit to (A^*, B^*) by computing $\mathbf{C}^*_A = \mathsf{Com}(\mathbf{u}, A^*)$, $\mathbf{C}^*_B = \mathsf{Com}(\mathbf{u}, B^*)$, then adapt the correspondingly simulated proof π^*_{ss}. As a side effect, π^*_{ss} is a valid proof of a false statement, where X is no longer equal to $1_{\mathbb{G}}$ as in Enc.

(d) For the proof π^*_{01}, since $\mathbf{C}^*_M = \mathsf{Com}(\mathbf{u}, 1_{\mathbb{G}})$, we update $\mathbf{C}_{\hat{M}^*} = \mathsf{Com}(\mathbf{v}, 1_{\hat{\mathbb{G}}})$. The Eq. 8 is valid as both sides of the equations are equal to 1. Then, compute the simulated proof π^*_{01} accordingly.

2. Randomizing CT^*
- Parse the CPA encryption part and randomize it as Rand at step 1a.
- Since $\bar{b} = 0$, recompute $\hat{H} = \hat{h}^{\bar{b}} = 1$ and $\hat{H}^\theta = 1$. Compute the corresponding commitments $\mathbf{C}^*_{\hat{H}} = \mathsf{Com}(\mathbf{v}, 1_{\hat{\mathbb{G}}})$ and $\mathbf{C}^*_\theta = \mathsf{Com}(\mathbf{v}, 1_{\hat{\mathbb{G}}})$. The verification Eqs. 2 and 3 are all valid since both sides are equal to 1. As a consequence, the encryption part is no more in the range of the honest CPA encryptions of $\mathsf{Dec}(\mathsf{SK}, C_b)$ except with probability $1/p$. Next, compute the proof π^*_b and π^*_θ as in Enc.

Game 3 and Game 4 abort in the same cases. When both games do not abort, their views are exactly the same thanks to the perfect witness indistinguishability of GS proofs. Particularly, the distributions of π^*_{ss} and randomized π'_{ss} are indistinguishable. We thus have $\Pr[S_4] = \Pr[S_3]$.

$\mathsf{Game}_5(\lambda)$**:** This game is as the previous game except that the Groth Sahai CRS $\mathbf{u}$ and $\mathbf{v}$ of the public key are now generated in the extractable mode. Namely,

we pick $\vec{u}_1 \xleftarrow{\$} \mathbb{G}^2$, $\gamma \xleftarrow{\$} \mathbb{Z}_p$, and compute $\vec{u}_2 = \vec{u}_1^{\gamma}$. The CRS forms a random DH tuple over $\mathbb{G}$. Thus, $|\Pr[S_5] - \Pr[S_4]| \leq 2\epsilon_{sxdh}$.

$\mathsf{Game}_6(\lambda)$**:** We bring the following modification to the previous game. When sampling CRS $\mathbf{u} = (\vec{u}_1, \vec{u}_1^{\gamma})$, we compute $\vec{u}_1 = (u_{11}, u_{12})$, where $u_{12} = u_{11}^{\mu}$ with $\mu \xleftarrow{\$} \mathbb{Z}_p$. As per [23], the distribution of the public key is unchanged, but we keep μ as an ElGamal secret key to extract the committed group elements of the Groth-Sahai commitments. Moreover, when receiving $\mathbf{C}_A^{(b)}, \mathbf{C}_B^{(b)}, \mathbf{C}_G^{(b)}$ from the adversary, we extract some $A^{(b)}, B^{(b)}, G^{(b)} \in \mathbb{G}$. Here, we introduce a failure event F_6 when $\mathcal{A}$ can produce a valid signature satisfying Eq. 7 when $G^{(b)} \neq g$ (and then $\hat{H}^{(b)} \neq \hat{h}$) in at least one of the following situations: in any pre-challenge decryption query, in the challenge phase with C_0 or C_1. In other words, we reject all the valid ciphertexts in the sense of Game 5 for which $\pi_{ss}^{(b)}$ is a valid proof for a false statement. As a result, we abort and output 0 if the adversary can successfully create a valid but dishonest signature $(A^{(b)}, B^{(b)})$ on a message different from (1, 1). We have $|\Pr[S_6] - \Pr[S_5]| \leq \Pr[F_6]$.
To compute $\Pr[F_6]$, let $(A^{\dagger}, B^{\dagger})$ the honest signature on $g/G^{(b)}$, $(A^{\dagger}, B^{\dagger}) = \mathtt{Sign}(\mathsf{sk}_{\mathsf{lhsp}}, (g/G^{(b)}, g/G^{(b)^{\tau}})$. There are 2 cases that F_6 can occur: (1) The adversary $\mathcal{A}$ can correctly guess $(A^{(b)}, B^{(b)}) = (A^{\dagger}, B^{\dagger})$ with a probability of $1/p$ or (2) $(A^{(b)}, B^{(b)}) \neq (A^{\dagger}, B^{\dagger})$ is a valid but dishonest signature on $(g/G^{(b)}, (g/G^{(b)})^{\tau})$. Considering the second case, we have both $(A^{\dagger}, B^{\dagger})$ and $(A^{(b)}, B^{(b)})$ satisfying Eq. 7 with the same right-hand side member. This implies an SXDH distinguisher. We thus have $\Pr[F_6] \leq 1/p + (1 - 1/p)\epsilon_{sxdh} \leq 1/p + \epsilon_{sxdh}$, therefore $|\Pr[S_6] - \Pr[S_5]| \leq 1/p + \epsilon_{sxdh}$.

$\mathsf{Game}_7(\lambda)$**:** This game is the same as Game 6 except that we introduce a failure event when $\mathcal{A}$ can produce a valid signature when $G^{(i)} \neq g$ in post-challenge decryption query with $\mathrm{Trace}(\mathsf{PK}, C^{(i)}) \neq \mathsf{opk}^*$. Similarly to the previous game, when receiving $\mathbf{C}_A^{(i)}, \mathbf{C}_B^{(i)}, \mathbf{C}_G^{(i)}$ from the adversary for a decryption query, we extract some $A^{(i)}, B^{(i)}, G^{(i)} \in \mathbb{G}$. Since $\mathcal{A}$ has to use a different tag $\tau \neq \tau^*$ for post-challenge decryption queries, the message $(X^{(i)}, X^{(i)^{\tau}}) = (g/G^{(i)}, g/G^{(i)^{\tau}})$ is not in $\mathrm{span}\langle (X^*, X^{*^{\tau^*}}) \rangle$. Thanks to the unforgeability of the LHSP signature, the validity of Eq. 7 implies trivial, when $X^{(i)} = 1$ and $G^{(i)} = g$. Hence, after observing a simulated proof π_{ss}^* for a false statement in Game 6, the adversary is not able to validate another falsely simulated proof for a false statement. Thus, $|\Pr[S_7] - \Pr[S_6]| \leq 1/p + \epsilon_{sxdh}$.

$\mathsf{Game}_8(\lambda)$**:** Up to this point, if the game does not abort, all the ciphertexts from an adversary can not contain a valid signature of a message different to $(1_{\mathbb{G}}, 1_{\mathbb{G}})$. That means all the ciphertexts that will be decrypted are honest and do not reveal any information of SK, except those provided in the challenge phase. In this game, we bring another modification in the way we generate the CPA encryption part. To make sure the challenge ciphertext C^* does not contain any information of which C_b is used in randomization, let us call $G_1 = g_1^{\theta^*} \in \mathbb{G}$, $H_1 = h_1^{\theta^*} \in \mathbb{G}$, since $f_1 = g_1^{\alpha_1} h_1^{\beta_1}$ we compute $F_1 = G_1^{\alpha_1} H_1^{\beta_1}$ using the secret key $\mathsf{SK} = (\alpha_1, \beta_1)$. (g_1, h_1, G_1, H_1) forms a random DDH tuple over $\mathbb{G}$. The challenge ciphertext in Game 4 is then $\mathbf{c}_m^* = (c_0^*, c_1^*, c_2^*) =$

$\mathbf{c}_m^{(b)} \cdot (f_1, g_1, h_1)^{\theta^*} = \mathbf{c}_m \cdot (F_1, G_1, H_1)$. Now, instead of choosing G_1, H_1 like this, we pick random $G_1, H_1 \xleftarrow{\$} \mathbb{G}$ and compute $F_1 = G_1^{\alpha_1} H_1^{\beta_1}$, the tuple (g_1, h_1, G_1, H_1) is a random quadruple in $\mathbb{G}$. As a result, $\mathbf{c}_m^* = (c_0^*, c_1^*, c_2^*) = \mathbf{c}_m \cdot (F_1, G_1, H_1)$ is no more in the range of the honest CPA encryptions of $\mathsf{Dec}(\mathsf{SK}, C_b)$ except with probability $1/p$. Consequently, π_θ^* is a proof of a false statement but valid since $\hat{H} = \hat{H}^{\theta^*} = 1$ as set in Game 4. Obviously, $|\Pr[S_8] - \Pr[S_7]| \leq \epsilon_{sxdh}$ since the distinction between them is the distinction between a random DDH tuple and a random quadruple in $\mathbb{G}$.

In fact, after observing the simulated proof π_θ^*, the adversary is not able to do the same, i.e., setting $\hat{H} = \hat{H}^{\theta^*} = 1$. Since π_b^* has to be valid, the soundness of GS proof shows that $(G, \hat{H})$ is in the form of $(g^{\bar{b}}, \hat{h}^{\bar{b}})$. However, $G^{(b)} = g$ because $G^{(b)} \neq g$ is aborted from Game 7. Therefore, $\bar{b} = 1$ and $\hat{H} = \hat{h} \neq 1$.

To conclude, we need to compute the $\Pr[S_8]$. Firstly, we argue that A's view in Game 8 is statistically independent of the hidden bit b. If the game aborts and outputs a random bit, the probability of returning 1 is 1/2. If there is no abort, that is, all the ciphertexts C for decryption queries are honest and $\mathsf{Dec}(\mathsf{SK}, C) = (c_0 \cdot c_1^{-\alpha_1} \cdot c_2^{-\beta_1})$ does not reveal any additional information about the secret key SK, except what can be inferred from $f_1 = g_1^{\alpha_1} h_1^{\beta_1}$ and $F_1 = G_1^{\alpha_1} H_1^{\beta_1}$, where G_1, H_1 are kept secret during the computation of the challenge ciphertext. Suppose that $G_1 = g_1^y$ and $H_1 = h_1^y f_1^z$ for random $y, z \xleftarrow{\$} \mathbb{Z}_p$, we have $F_1 = f_1^{y+z\beta}$. As a consequence, the computation of $\mathbf{c}_m^* = \mathbf{c}_m^{(b)} \cdot (F_1, G_1, H_1) = (c_0^{(b)} \cdot f_1^{y+z\beta}, c_1^{(b)} \cdot g_1^y, c_2^{(b)} \cdot h_1^y f_1^z)$. If at least one of the two values (y, z) is 0, the probability that $\mathcal{A}$ wins is $P_1 \leq 2/p + 1/p^2$. If both $y, z \neq 0$, $\mathbf{c}_m^*$ is a random triple over $\mathbb{G}^3$, $\mathcal{A}$ wins with the probability of $P_2 = 1/2(1 - 2/p - 1/p^2)$. Finally, the probability that $\mathcal{A}$ wins in this game is $\Pr[S_8] \leq P_1 + P_2 \leq 1/2 + 2/p$.

In summary, we have $\left|\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathrm{tcca}}(\lambda) = 1] - \frac{1}{2}\right| \leq \epsilon_{cr} + 6\epsilon_{sxdh} + \frac{4}{p}$.

B.4 Traceability

Theorem 3.3. *The above scheme is traceable (Fig. 1) under the SXDH assumption. More precisely, for any adversary $\mathcal{A}$, we have* $\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathrm{trace}}(\lambda) = 1] \leq 5\epsilon_{sxdh} + \frac{1}{p}$.

Proof. Let $\mathcal{A}$ be an efficient adversary against the traceability of our scheme. 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 the game output 1.

$\mathsf{Game}_1(\lambda)$**:** This is the real game as described in the experiment Fig. 1, where $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{Gen}(1^\lambda)$. Then, $(m, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{PK}, \mathsf{SK})$, $C = (CT, D) \leftarrow \mathrm{Enc}(\mathsf{PK}, m)$, and $C^* = (CT^*, D^*) \leftarrow \mathcal{A}_2(\mathsf{st}, C)$. By definition, S_1 occurs if $\mathrm{Ver}(\mathsf{PK}, C^*) = 1$, $\mathrm{Dec}(\mathsf{SK}, C^*) \neq m$, and $\mathsf{opk}^* = \mathrm{Trace}(\mathsf{PK}, C^*) = \mathrm{Trace}(\mathsf{PK}, C) = \mathsf{opk}$. Thus, $\Pr[S_1] = \Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathrm{trace}}(\lambda) = 1]$.

$\mathsf{Game}_2(\lambda)$**:** This game is as the real game except that the Groth Sahai CRSes $\mathbf{u} = (\vec{u}_1, \vec{u}_2) \in \mathbb{G}^4$ and $\mathbf{v} = (\vec{v}_1, \vec{v}_2), \mathbf{v}' = (\vec{v}'_1, \vec{v}'_2) \in \hat{\mathbb{G}}^4$ of the public key are now generated in the extractable mode. In particular, instead of picking them uniformly at random, we pick them as random Diffie-Hellman tuples over the appropriate groups. Under the DDH assumptions in $\mathbb{G}$ and $\hat{\mathbb{G}}$, the adversary does not notice the difference. Thus, any adversary's behavior to distinguish between Game 1 and Game 2 leads to a SXDH distinguisher. That means $|\Pr[S_1] - \Pr[S_2]| \leq 3\epsilon_{sxdh}$.

$\mathsf{Game}_3(\lambda)$**:** We introduce one more modification to Game 2 in the way to generate the commitment key $\hat{g}_1, \hat{h}_1, \hat{g}_2, \hat{h}_2$ of PK. Instead of picking them all uniformly over $\hat{\mathbb{G}}$, we pick a random scalar $x \xleftarrow{\$} \mathbb{Z}_p$ and set $(\hat{g}_2, \hat{h}_2) = (\hat{g}_1, \hat{h}_1)^x$. This modification turns the perfectly hiding commitment $(\hat{d}_1, \hat{d}_2) = (\hat{g}^m, 1) \cdot (\hat{g}_1, \hat{g}_2)^r \cdot (\hat{h}_1, \hat{h}_2)^q$ into an extractable commitment $(\hat{g}^m \hat{g}_1^r \hat{h}_1^q, (\hat{g}_1^r \hat{h}_1^q)^x)$. Moreover, the last two lines of the matrix T in Eq. (5) are now linearly dependent, so that the row space of T is now a 2-dimensional sub-space over $\hat{\mathbb{G}}^3$. By the SXDH assumption, we have $|\Pr[S_2] - \Pr[S_3]| \leq \epsilon_{sxdh}$.

$\mathsf{Game}_4(\lambda)$**:** This game is the same as the previous game except that we introduce a failure event, which causes the game to be aborted and output 0. When we generate $C \leftarrow \mathsf{Enc}(\mathsf{PK}, m)$ given m from $\mathcal{A}_1$, we first compute $(\mathsf{opk}, \mathsf{osk}) \leftarrow \mathsf{LGen}(\mathsf{PK})$ and then $C \leftarrow \mathsf{LEnc}(\mathsf{PK}, \mathsf{osk}, m)$ as before, but we keep osk. Then, as soon as we get C^* from $\mathcal{A}_2$ with the commitment $(\hat{d}_1^*, \hat{d}_2^*)$, we extract the necessarily valid $\hat{\sigma}_1^* = (\hat{Z}^*, \hat{R}^*)$ LHSP signature from the (now perfectly sound) GS proof and compare it to $\hat{\sigma}_1^\dagger = \mathsf{Sign}(\mathsf{osk}, (\hat{g}, \hat{d}_1^*, \hat{d}_2^*))$. The failure event happens if $\hat{\sigma}_1^* \neq \hat{\sigma}_1^\dagger$. Due to the property of the LHSP signature [28], if we have two distinct signatures on a same vector we can solve the DDH problem. We thus have $|\Pr[S_3] - \Pr[S_4]| \leq \epsilon_{sxdh}$.

We conclude by showing that $\Pr[S_4] = 1/p$. Indeed, S_4 is an event when $\mathcal{A}$ wins by correctly guessing $\hat{\sigma}_1^* = \mathsf{Sign}(\mathsf{osk}, (\hat{g}, \hat{d}_1^*, \hat{d}_2^*))$, but $m \neq \mathsf{Dec}(\mathsf{SK}, C^*)$. That is, $(\hat{g}, \hat{d}_1^*, \hat{d}_2^*)$ is not in the 2-dimensional linear span of the row vectors of T signed in C. Since osk contains enough entropy after C was given to the adversary, $Z^\dagger$ is still unknown and uniform over $\mathbb{G}$. Therefore the probability to have $\hat{Z}^* = \hat{Z}$ is $1/p$.

In summary, we have $\Pr[\mathsf{Exp}^{\mathrm{trace}}_{\mathcal{A}}(\lambda) = 1] \leq 5\epsilon_{sxdh} + \frac{1}{p}$.

B.5 Verifiability

Theorem 3.4. *The above TREnc is verifiable under the SXDH assumption. More precisely, for any adversary $\mathcal{A}$, we have* $\Pr[\mathsf{Exp}^{\mathrm{ver}}_{\mathcal{A}}(\lambda) = 1] \leq 3\epsilon_{sxdh} + \frac{1}{p}$.

Proof. Given $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{Gen}(1^\lambda)$, we have to show that any ciphertext from $\mathcal{A}$ which passes the verification equations is necessarily in the range of the honestly generated encryptions with overwhelming probability. In other words, $\Pr[\mathsf{Exp}^{\mathrm{ver}}_{\mathcal{A}}(\lambda) = 1]$ is defined that if $C \leftarrow \mathcal{A}(\mathsf{PK}, \mathsf{SK})$ is not in the honest encryption range, the probability that it is considered as valid is negligible.

Let $C = (CT, D) \leftarrow \mathcal{A}(\mathsf{PK}, \mathsf{SK})$ satisfying $\mathsf{Ver}(\mathsf{PK}, C) = 1$, where $CT = (\mathbf{c}_m, c_r, c_q, \mathbf{C}_{\hat{H}}, \mathbf{C}_\theta, \pi_b, \pi_\theta)$ and $D = (\hat{d}_1, \hat{d}_2, \mathbf{C}_{\hat{M}}, \mathbf{C}_M, \mathbf{C}_R, \mathbf{C}_Q, \mathbf{C}_G, \mathbf{C}_{\hat{Z}}, \mathbf{C}_{\hat{R}}, \mathbf{C}_A, \mathbf{C}_B, \pi_{open}, \hat{\sigma}_2, \hat{\sigma}_3, \pi_{sig}, \pi_{ss}, \pi_{01}, \mathsf{opk})$.

To show that the CPA part of CT is well formed, we rely on the soundness of the proof related to the CRS $\mathbf{u}, \mathbf{v}$. As in the TCCA proof, we switch these CRSes to the extractable mode, which leads to a security loss of $2\epsilon_{sxdh}$. Next, we extract a witness from the valid proofs associated with $\mathbf{u}, \mathbf{v}$. If $(A, B, X) \neq (1, 1, 1)$, we abort. That is, the adversary manages to produce a valid LHSP signature for the public key $(\hat{f}_1, \hat{f}_2)$. By generating this pair in the key generation so that we know a corresponding secret key, we can show that this happens with negligible probability $\epsilon_{sxdh} + 1/p$ from the LHSP unforgeability. From now on, we can thus assume that the extracted $G = g$, $\hat{H} = \hat{h}$. Therefore, the soundness of GS proofs allows extracting non-trivial witness from the satisfiability of Eq. (3), which shows that $\mathbf{c}_m = (c_0, c_1, c_2)$, c_r and c_q have the expected honest structure.

The LHSP signatures and the GS proof associated with the CRS $\mathbf{v}'$ can always be explained honestly, even if it is not efficient to compute their discrete log representation. The same happens for the perfectly hiding commitment $(\hat{d}_1, \hat{d}_2)$ since we can extract the opening in Eq. (4) with respect to $\mathbf{u}, \mathbf{v}$, which must be consistent with decryption of (M, R, Q). Moreover, $M = g^m$ must be a bit thanks to Eq. (8).

To conclude, we have $\Pr[\mathsf{Exp}^{\mathrm{ver}}_{\mathcal{A}}(\lambda) = 1] \leq 3\epsilon_{sxdh} + \frac{1}{p}$.

References

1. Adida, B.: Helios: web-based open-audit voting. In: Proceedings of the 17th USENIX Security Symposium, pp. 335–348. USENIX Association (2008)
2. 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)
3. Barreto, P.S.L.M., Naehrig, M.: Pairing-friendly elliptic curves of prime order. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 319–331. Springer, Heidelberg (2006). https://doi.org/10.1007/11693383_22
4. Benaloh, J., Naehrig, M.: Electionguard design specification version 2.0.0. https://www.electionguard.vote/spec/. Accessed Aug 2023
5. Benaloh, J., Tuinstra, D.: Receipt-free secret-ballot elections. In: Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, pp. 544–553 (1994)
6. 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, SP 2015, San Jose, CA, USA, 17–21 May 2015, pp. 499–516. IEEE Computer Society (2015). https://doi.org/10.1109/SP.2015.37
7. Bernhard, D., Cortier, V., Pereira, O., Smyth, B., Warinschi, B.: Adapting Helios for provable ballot privacy. In: Atluri, V., Diaz, C. (eds.) ESORICS 2011. LNCS, vol. 6879, pp. 335–354. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23822-2_19

8. Bernhard, D., Pereira, O., Warinschi, B.: On necessary and sufficient conditions for private ballot submission. Cryptology ePrint Archive (2012)
9. Blazy, O., Fuchsbauer, G., Pointcheval, D., Vergnaud, D.: Signatures on randomizable ciphertexts. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 403–422. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19379-8_25
10. Boneh, D.: The decision Diffie-Hellman problem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 48–63. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0054851
11. Canard, S., Schoenmakers, B., Stam, M., Traoré, J.: List signature schemes. Discret. Appl. Math. **154**(2), 189–201 (2006)
12. 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, pp. 1614–1625 (2016)
13. Chaum, D., et al.: Scantegrity II: end-to-end verifiability by voters of optical scan elections through confirmation codes. IEEE Trans. Inf. Forensics Secur. **4**(4), 611–627 (2009)
14. Chaum, D., Ryan, P.Y.A., Schneider, S.: A practical voter-verifiable election scheme. In: di Vimercati, S.C., Syverson, P., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 118–139. Springer, Heidelberg (2005). https://doi.org/10.1007/11555827_8
15. Cortier, V., Gaudry, P., Glondu, S.: Belenios: a simple private and verifiable electronic voting system. In: Guttman, J.D., Landwehr, C.E., Meseguer, J., Pavlovic, D. (eds.) Foundations of Security, Protocols, and Equational Reasoning. LNCS, vol. 11565, pp. 214–238. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-19052-1_14
16. Cramer, R., Franklin, M., Schoenmakers, B., Yung, M.: Multi-authority secret-ballot elections with linear work. In: Maurer, U. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 72–83. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-68339-9_7
17. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. Eur. Trans. Telecommun. **8**(5), 481–490 (1997)
18. Culnane, C., Ryan, P.Y.A., Schneider, S.A., Teague, V.: vvote: A verifiable voting system. ACM Trans. Inf. Syst. Secur. **18**(1), 3:1–3:30 (2015)
19. Cuvelier, É., Pereira, O., Peters, T.: Election verifiability or ballot privacy: do we need to choose? In: Crampton, J., Jajodia, S., Mayes, K. (eds.) ESORICS 2013. LNCS, vol. 8134, pp. 481–498. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40203-6_27
20. Devillez, H., Pereira, O., Peters, T.: Traceable receipt-free encryption. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, Part III. LNCS, vol. 13793, pp. 273–303. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-22969-5_10
21. Grewal, G.S., Ryan, M.D., Bursuc, S., Ryan, P.Y.A.: Caveat coercitor: coercion-evidence in electronic voting. In: 2013 IEEE Symposium on Security and Privacy, SP 2013, pp. 367–381. IEEE Computer Society (2013)
22. Grontas, P., Pagourtzis, A., Zacharakis, A., Zhang, B.: Towards everlasting privacy and efficient coercion resistance in remote electronic voting. In: Zohar, A., et al. (eds.) FC 2018. LNCS, vol. 10958, pp. 210–231. Springer, Heidelberg (2019). https://doi.org/10.1007/978-3-662-58820-8_15
23. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_24

24. Haines, T., Mueller, J., Mosaheb, R., Pryvalov, I.: SoK: secure e-voting with everlasting privacy. In: Proceedings on Privacy Enhancing Technologies (PoPETs) (2023)
25. Hirt, M., Sako, K.: Efficient receipt-free voting based on homomorphic encryption. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 539–556. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45539-6_38
26. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society, pp. 61–70 (2005)
27. Kiltz, E., Wee, H.: Quasi-adaptive NIZK for linear subspaces revisited. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 101–128. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_4
28. Libert, B., Peters, T., Joye, M., Yung, M.: Linearly homomorphic structure-preserving signatures and their applications. Des. Codes Crypt. **77**, 441–477 (2015)
29. Libert, B., Peters, T., Qian, C.: Structure-preserving chosen-ciphertext security with shorter verifiable ciphertexts. In: Fehr, S. (ed.) PKC 2017. LNCS, vol. 10174, pp. 247–276. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54365-8_11
30. Locher, P., Haenni, R.: Receipt-free remote electronic elections with everlasting privacy. Ann. Telecommun. **71**, 323–336 (2016)
31. The miraCL core cryptographic library. https://github.com/miracl/core
32. Moran, T., Naor, M.: Receipt-free universally-verifiable voting with everlasting privacy. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 373–392. Springer, Heidelberg (2006). https://doi.org/10.1007/11818175_22
33. Okamoto, T.: Receipt-free electronic voting schemes for large scale elections. In: Christianson, B., Crispo, B., Lomas, M., Roe, M. (eds.) Security Protocols 1997. LNCS, vol. 1361, pp. 25–35. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0028157
34. Ryan, P.Y.A., Rønne, P.B., Iovino, V.: Selene: voting with transparent verifiability and coercion-mitigation. In: Clark, J., Meiklejohn, S., Ryan, P.Y.A., Wallach, D., Brenner, M., Rohloff, K. (eds.) FC 2016. LNCS, vol. 9604, pp. 176–192. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53357-4_12
35. Sako, K., Kilian, J.: Receipt-free mix-type voting scheme: a practical solution to the implementation of a voting booth. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 393–403. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-49264-X_32
36. Wikström, D.: Verificatum. https://www.verificatum.org/. Accessed May 2022