Mix-Nets from Re-Randomizable and Replayable CCA-secure Public-Key Encryption

Antonio Faonio and Luigi Russo

EURECOM, Sophia Antipolis, France
{faonio, russol}@eurecom.fr

Abstract. Mix-nets are protocols that allow a set of senders to send messages anonymously. Faonio *et al.* (ASIACRYPT'19) showed how to instantiate mix-net protocols based on Public-Verifiable Re-randomizable Replayable CCA-secure (Rand-RCCA) PKE schemes. The bottleneck of their approach is that public-verifiable Rand-RCCA PKEs are less efficient than typical CPA-secure re-randomizable PKEs. In this paper, we revisit their mix-net protocol, showing how to get rid of the cumbersome public-verifiability property, and we give a more efficient instantiation for the mix-net protocol based on a (non publicly-verifiable) Rand-RCCA scheme. Additionally, we give a more careful security analysis of their mix-net protocol.

1 Introduction

Mixing Networks (aka mix-nets), originally proposed by Chaum [Cha81], are protocols that allow a set of senders to send messages anonymously. Typically, a mix-net is realized by a chain of mix-servers (aka mixers) that work as follows. Senders encrypt their messages and send the ciphertexts to the first mix-server in the chain; each mix-server applies a transformation to every ciphertext (e.g., partial decryption, or re-encryption), re-orders the ciphertexts according to a secret random permutation, and passes the new list to the next mix-server. The idea is that the list returned by the last mixer contains (either in clear or encrypted form, depending on the mixing approach) the messages sent by the senders in a randomly permuted order.

Mix-net protocols are fundamental building blocks to achieve privacy in a variety of application scenarios, including anonymous e-mail [Cha81], anonymous payments [JM99], and electronic voting [Cha81]. Informally, the basic security property of mix-nets asks that, when enough mix-servers are honest, the privacy of the senders of the messages (i.e., "who sent what") is preserved. In several applications, it is also desirable to achieve correctness even in the presence of an arbitrary number of dishonest mixers. This is for example fundamental in electronic voting where a dishonest mixer could replace all the ciphertexts with encrypted votes for the desired candidate.

Realizing Mix-Nets. A popular design paradigm of mixing networks are *re-encryption mix-nets* [PIK94] in which each server decrypts and freshly encrypts every ciphertext. Interestingly, such a transformation can be computed even publicly using re-randomizable encryption schemes (e.g., El Gamal). The process of re-randomizing and randomly permuting ciphertexts is typically called a *shuffle*. Although shuffle-based mix-nets achieve privacy when all the mix-servers behave honestly, they become insecure if one or more mixers do not follow the protocol. An elegant approach proposed to solve this problem is to let each mixer prove the correctness of its shuffle with a zero-knowledge proof. This idea inspired a long series of works on zero-knowledge shuffle arguments, e.g., [BG12,FS01,Gro03,Gro10,Nef01,TW10,Wik05,Wik09]. Notably, some recent works [BG12,TW10,Wik09] improved significantly over the early solutions, and they have been implemented and tested in real-world applications (elections) [Wik10]. In spite of the last results, zero-knowledge shuffle arguments are still a major source of inefficiency in mix-nets. This is especially a concern in applications like electronic voting where mix-nets need to be able to scale up to millions of senders (i.e., voters).

Mix-Nets from Replayable CCA Security. Most of the research effort for improving the efficiency of mix-nets has been so far devoted to improving the efficiency of shuffle arguments. A notable exception is the work of Faonio *et al.* [FFHR19]. Typical mixing networks based on re-randomizable encryption schemes make use of public-key encryption (PKE) schemes that are secure against chosen-plaintext attack (CPA), thus to obtain security against malicious mixers they leverage on the strong integrity property offered by the zero-knowledge shuffle arguments. The work of Faonio *et al.* instead showed that, by requiring stronger security properties from the re-randomizable encryption scheme, the NP-relation proved by the zero-knowledge shuffle arguments can be relaxed. This design enables faster and more efficient instantiations for the zero-knowledge proof but, on the other hand, requires more complex ciphertexts and thus a re-randomization procedure that is slower in comparison, for example, with the re-randomization procedure for ElGamal ciphertexts. More in detail, Faonio *et al.* propose a secure mixing network in the universal composability model of Canetti [Can01] based on re-randomizable PKE schemes that are replayable-CCA (RCCA) secure (as defined by Canetti *et al.* [CKN03]) and publicly-verifiable. The first notion, namely RCCA security, is a relaxation of the standard notion of chosen-ciphertext security. This notion offers security against malleability attacks on the encrypted message (i.e. an attacker cannot *transform* a ciphertext of a message $\mathtt{M}$ to a ciphertext of a message $\mathtt{M}'$) but it still allows for malleability on the ciphertext (i.e. we can re-randomize the ciphertexts). The second requirement, namely public verifiability, requires that anyone in possession of the public key can check that a ciphertext decrypts correctly to a valid message, in other words, that the decryption procedure would not output an error message on input such a ciphertext. Unfortunately, this second requirement is the source of the major inefficiency in the mixing networks of Faonio *et al.*. For example, the re-randomization procedure of the state-of-art non publicly-verifiable re-randomizable PKE scheme with RCCA-security (Rand-RCCA PKE, in brief) in the random oracle model of Faonio and Fiore [FF20] costs 19 exponentiations in a pairing-free cryptographic group, while the re-randomization procedure of the publicly-verifiable Rand-RCCA PKE of [FFHR19] costs around 90 exponentiations plus 5 pairing operations.

1.1 Our Contribution

We revisit the mix-net design of Faonio *et al.* [FFHR19]. Our contributions are two-fold: we generalize the mix-net protocol of [FFHR19] showing how to get rid of the cumbersome public verifiability property, and we give a more efficient instantiation for the mix-net protocol based on the (non publicly-verifiable) Rand-RCCA scheme of [FFHR19]. Our generalization of the mix-net protocol is based on two main ideas. The first idea is that, although the verification of the ciphertexts is still necessary, it is not critical for the verification to be public and non-interactive. In particular, we can replace the public verifiability property with a multi-party protocol (that we call a *verify-then-decrypt* protocol) that verifies the ciphertexts before the decryption phase and that decrypts the ciphertexts from the last mixer in the chain only if the verification succeeded. The second idea is that in the design of the *verify-then-decrypt* multiparty protocol we can trade efficiency for security. In particular, we could design a protocol that eventually leaks partial information about the secret key and, if the Rand-RCCA PKE scheme is resilient against this partial leakage of the secret key, we could still obtain a secure mix-net protocol. Along the way, we additionally (1) abstract the necessary properties required by the zero-knowledge proof that the mixers need to attach to their shuffled ciphertexts and (2) give a more careful security analysis of the mixnet protocol. More technically, we define the notion *sumcheck-admissible relation* w.r.t. the Rand-RCCA PKE scheme (see Definition 5) which is a property of an NP-relation that, informally, states that given two lists of ciphertexts if all the ciphertexts in the lists decrypt to valid messages, then the sum of the messages in the first list is equal to the sum of the messages in the second list. For example, a shuffle relation is a sumcheck-admissible relation, however simpler (and easier to realize in zero-knowledge) NP-relations over the lists of ciphertexts can be considered as well.

Our second contribution is a concrete instantiation of the mix-net protocol. The main idea of our concrete protocol is that many (R)CCA PKE schemes can be conceptually divided into two main components: the first *"CPA-secure"* component assures that the messages are kept private, while the second component assures the integrity of the ciphertexts, namely, the component can identify malformed ciphertexts and avoid dangerous decryptions through the CPA-secure component. Typical examples for such PKE schemes are those based on the Cramer-Shoup paradigm [CS02]. Intuitively, these schemes should be secure even if the adversary gets to see the secret key associated with the second component under the constraint that once such leakage is available the adversary must lose access to the decryption oracle. This suggests a very efficient design for the *verify-then-decrypt* multiparty protocol: the mixers commit to secret shares of the secret key, once all the ciphertexts are available the mixers open to the secret key material for the second component, now any mixer can non-interactively and efficiently verify the validity of the ciphertexts. If all the ciphertexts are valid the mixers can engage a CPA-decryption multiparty protocol for the ciphertexts in the last list. As last contribution, we show that the Rand-RCCA PKE scheme of [FFHR19] is leakage resilient (under the aforementioned notion) and we instantiate all the necessary parts.

A final remark, an important property of a mixnet protocol is the so-called *auditability*[1], namely the capability of an external party to verify that a given transcript of a protocol execution has produced an alleged output. Intuitively, mixnets based on non-interactive zero-knowledge proofs of shuffle usually should have this property. However, one must be careful, because not only the shuffling phase, but the full mixnet protocol should be auditable. In particular, for our mixnet protocol to be auditable the verify-then-decrypt protocol should be auditable as well. We show that the latter protocol for our concrete instantiation is indeed auditable.

1.2 Related work

The notion of mix-net was introduced by Chaum [Cha81]. The use of zero-knowledge arguments to prove the correctness of a shuffle was first suggested by Sako and Kilian [SK95]. The first proposals used expensive *cut-and-choose*-based zero-knowledge techniques [Abe98,SK95]. Abe *et al.* removed the need for cut-and-choose by proposing a shuffle based on permutation networks [Abe99,AH01]. Furukawa and Sako [FS01] and independently Neff [Nef01] proposed the first zero-knowledge shuffle arguments for ElGamal ciphertexts that achieve a complexity linear in the number of ciphertexts. These results have been improved by Wikström [Wik09], and later Terelius and Wikström [TW10], who proposed arguments where the proof generation can be split into an offline and online phase (based on an idea of Adida and Wikström [AW07]). These protocols have been implemented in the Verificatum library [Wik10]. Groth and Ishai [GI08] proposed the first zero-knowledge shuffle argument with sublinear communication. Bayer and Groth gave a faster argument with sublinear communication in [BG12]. The notion of Rand-RCCA PKE encryption was introduced by Groth [Gro04]. The work of Prabhakaran and Rosulek [PR07] showed the first Rand-RCCA PKE in the standard model. The work of Faonio and Fiore [FF20] presented a practical Rand-RCCA PKE scheme in the random oracle model. Recently, Wang *et al.* [WCY+21] introduced the first receiver-anonymous Rand-RCCA PKE, solving the open problem raised by Prabhakaran and Rosulek in [PR07]. The state-of-art Rand-RCCA PKE scheme can be found in the work of Faonio *et al.* [FFHR19]. Other publicly-verifiable Rand-RCCA PKE schemes were presented by Chase *et al.* [CKLM12] and Libert *et al.* [LPQ17]. As far as we know, our design for the *verify-then-decrypt* protocol cannot be easily instantiated with the schemes in [FF20,PR07,WCY+21]. The reason is that for all these schemes the decryption procedures have a "verification step" that depends on the encrypted message.

[1] This notion is sometimes called *verifiability*, however, we prefer to use the term "auditability" to avoid confusion with the verifiability of the ciphertexts property.

2 Preliminaries

A function is negligible in λ if it vanishes faster than the inverse of any polynomial in λ. We write $f(\lambda) \in \mathsf{negl}(\lambda)$ when f is negligible in λ.

The expression $[n]$ denotes the set $\{1, 2, \ldots, n\}$ for an integer $n \geq 1$. Calligraphic letters denote the sets, while set sizes are written as $|\mathcal{X}|$. Capital letters denote the lists; they are represented as ordered tuples, e.g. $L \coloneqq (L_i)_{i\in[n]}$ is a shortcut for the list of n elements $(L_1, \ldots, L_n)$. Given n lists $L_i, i \in [n]$, and an element x, we define the following operations: (i) $\mathsf{Count}(x, L_i)$ returns the number of times the value x appears in the list L_i, (ii) $\mathsf{Concat}(L_1, \ldots, L_n)$ returns a list L as a concatenation of the input lists, and $L_1 \subseteq L_2$ returns 1 if each element of L_1 is contained in the list L_2, or 0 otherwise.

An asymmetric bilinear group $\mathcal{G}$ is a tuple $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathcal{P}_1, \mathcal{P}_2)$, where $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are groups of prime order q, the elements $\mathcal{P}_1, \mathcal{P}_2$ are generators of $\mathbb{G}_1, \mathbb{G}_2$ respectively, $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently-computable non-degenerate bilinear map, and there is no efficiently computable isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$. Let GGen be some probabilistic polynomial-time algorithm which on input 1^λ, where λ is the security parameter, returns a description of a bilinear group $\mathcal{G}$. Elements in $\mathbb{G}_i, i \in \{1, 2, T\}$ are denoted in implicit notation as $[a]_i \coloneqq a\mathcal{P}_i$, where $\mathcal{P}_T \coloneqq e(\mathcal{P}_1, \mathcal{P}_2)$. Every element in $\mathbb{G}_i$ can be written as $[a]_i$ for some $a \in \mathbb{Z}_q$, but note that given $[a]_i$, $a \in \mathbb{Z}_q$ is in general hard to compute (discrete logarithm problem). Given $a, b \in \mathbb{Z}_q$ we distinguish between $[ab]_i$, namely the group element whose discrete logarithm base $\mathcal{P}_i$ is ab, and $[a]_i \cdot b$, namely the execution of the multiplication of $[a]_i$ and b, and $[a]_1 \cdot [b]_1 = [a \cdot b]_T$, namely the execution of a pairing between $[a]_1$ and $[b]_1$.

Vectors and matrices are denoted in boldface. We extend the pairing operation to vectors and matrices as $e([\mathbf{A}]_1, [\mathbf{B}]_2) = [\mathbf{A}^\top \cdot \mathbf{B}]_T$ and $e([y]_1, [\mathbf{A}]_2) = [y \cdot \mathbf{A}]_T$.

We recall that a NP-relation $\mathcal{R}$ is a set of tuples (x, w) where x is the instance and w witness such that there exists an efficiently computable predicate R that decides the membership in the set, i.e. $R(x, w)$ outputs 1 if $(x, w) \in \mathcal{R}$. The corresponding language $\mathcal{L}(\mathcal{R})$ is the set of x for which there exists a witness w such that $(x, w) \in \mathcal{R}$.

2.1 Re-randomizable PKE

A re-randomizable PKE (Rand-PKE) scheme PKE is a tuple of five algorithms:

$\mathsf{Setup}(1^\lambda)$: upon input the security parameter 1^λ produces parameters prm, which include the description of the message and ciphertext space $\mathcal{M}$, $\mathcal{C}$.

$\mathsf{KGen}(\mathsf{prm})$: upon input the parameters prm, outputs a key pair $(\mathsf{pk}, \mathsf{sk})$.

$\mathsf{Enc}(\mathsf{pk}, \mathtt{M})$: upon inputs a public key pk and a message $\mathtt{M} \in \mathcal{M}$, outputs a ciphertext $\mathtt{C} \in \mathcal{C}$.

$\mathsf{Dec}(\mathsf{pk}, \mathsf{sk}, \mathtt{C})$: upon inputs a secret key sk and a ciphertext $\mathtt{C}$, outputs a message $\mathtt{M} \in \mathcal{M}$ or an error symbol $\perp$.

$\mathsf{Rand}(\mathsf{pk}, \mathtt{C})$: upon inputs a public key pk and a ciphertext C, outputs another ciphertext $\mathtt{C}'$.

Definition 1 (Perfect Re-randomizability, [FFHR19]). *We say that* PKE *is perfectly re-randomizable (Re-Rand, for short) if the following three conditions are met:*

(Indistinguishability) *For any $\lambda \in \mathbb{N}$, any $\mathsf{prm} \leftarrow\$ \mathsf{Setup}(1^\lambda)$, any $(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{KGen}(\mathsf{prm}, 1^\lambda)$, for any $\mathtt{M} \in \mathcal{M}$ and any $\mathtt{C} \in \mathsf{Enc}(\mathsf{pk}, \mathtt{M})$ the following two distributions are identical*

$$\mathtt{C}_0 \leftarrow\$ \mathsf{Enc}(\mathsf{pk}, \mathtt{M}) \text{ and } \mathtt{C}_1 \leftarrow\$ \mathsf{Rand}(\mathsf{pk}, \mathtt{C});$$

(Correctness) *For any $\lambda \in \mathbb{N}$, any $\mathsf{prm} \leftarrow\$ \mathsf{Setup}(1^\lambda)$, any $(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{KGen}(\mathsf{prm}, 1^\lambda)$, for any (possibly malicious) ciphertext C and every $\mathtt{C}' \leftarrow\$ \mathsf{Rand}(\mathsf{pk}, \mathtt{C})$ it holds*

$$\mathsf{Dec}(\mathsf{sk}, \mathtt{C}') = \mathsf{Dec}(\mathsf{sk}, \mathtt{C}).$$

(Tightness of Decryption) *For any (possibly unbounded) adversary $\mathcal{A}$ and any sequence of parameters $\{\mathsf{prm}_\lambda \leftarrow\$ \mathsf{Setup}(1^\lambda)\}_{\lambda \in \mathbb{N}}$ the following holds:*

$$\Pr\left[\exists \mathtt{M} : \mathtt{C} \notin \mathsf{Enc}(\mathsf{pk}, \mathtt{M}) \wedge \mathsf{Dec}(\mathsf{sk}, \mathtt{C}) = \mathtt{M} \neq \bot : \begin{array}{c}(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{KGen}(\mathsf{prm}_\lambda) \\ \mathtt{C} \leftarrow\$ \mathcal{A}(\mathsf{pk})\end{array}\right] \in \mathsf{negl}(\lambda).$$

2.2 All-but-One tag-based NIZK systems

Let $\mathsf{NIZK} = (\mathsf{Init}, \mathsf{P}, \mathsf{V})$ be a NIZK proof system for a relation $\mathcal{R}$ with tag space $\mathcal{T}$, and let $\mathsf{TPInit}(\mathsf{prm}, \tau)$ be an algorithm that upon input prm and a tag $\tau \in \mathcal{T}$ outputs a common reference string crs and trapdoor information $(\mathsf{tpe}, \mathsf{tps})$. Let $\tau \in \mathcal{T}$, and $\mathcal{T}'$ subset of $\mathcal{T}$. We define the following three properties:

- We say that TPInit is CRS indistinguishable w.r.t. NIZK if the common reference string generated by $\mathsf{Init}(\mathsf{prm})$ and the one generated by $\mathsf{TPInit}(\mathsf{prm}, \tau)$ are computationally indistinguishable, i.e. if for any sequences of $\{\mathsf{prm}_\lambda \leftarrow \mathsf{Setup}(1^\lambda)\}_{\lambda \in \mathbb{N}}$, and for any PT adversary $\mathcal{A}$:

$$|\Pr[\mathcal{A}(\mathsf{crs}) = 1 : \mathsf{crs} \leftarrow\$ \mathsf{Init}(\mathsf{prm}_\lambda)] - \Pr[\mathcal{A}(\mathsf{crs}) = 1 : \mathsf{crs}, \mathsf{tpe}, \mathsf{tps} \leftarrow\$ \mathsf{TPInit}(\mathsf{prm}_\lambda)]| \in \mathsf{negl}(\lambda)$$

- We say that NIZK is $\mathcal{T}'$-tag Composable Perfect Zero-Knowledge if there exists a PT TPInit that is CRS indistinguishable w.r.t. NIZK and for any prm, τ and any $(\mathsf{crs}, \mathsf{tpe}, \mathsf{tps}) \leftarrow \mathsf{TPInit}(\mathsf{prm}, \tau)$, and any $(x, w) \in \mathcal{R}$ and any $\tau' \in \mathcal{T}'$ we have that the distributions $\mathsf{P}(\mathsf{crs}, \tau', x, w)$ and $\mathsf{Sim}(\mathsf{tps}, \tau', x)$ are equivalently distributed.
- We say that NIZK is $\mathcal{T}'$-tag Adaptive Perfect g-Extractable if if there exists a PT TPInit that is CRS indistinguishable w.r.t. NIZK and for any prm, τ there exists a PT extractor Ext such that for any prm, τ, for any $(\mathsf{crs}, \mathsf{tps}, \mathsf{tpe}) \leftarrow \mathsf{TPInit}(\mathsf{prm}, \tau)$, and for any (possibly unbounded) adversary $(\tau', x, \pi) \leftarrow \mathcal{A}(\mathsf{crs})$ we have that if $\tau' \in \mathcal{T}'$ and $\mathsf{V}(\tau_{\text{snd}}, x, \pi) = 1$ then $\mathsf{Ext}(\mathsf{tpe}, \tau_{\text{snd}}, x, \pi)$ outputs z such that $\exists w : (x, w) \in \mathcal{R} \wedge g(w) = z$.

– We say that NIZK is $\mathcal{T}'$-tag Adaptive Perfect Sound if if there exists a PT TPInit that is CRS indistinguishable w.r.t. NIZK and for any prm, τ, for any prm, τ, for any $(\mathsf{crs}, \mathsf{tps}, \mathsf{tpe}) \leftarrow \mathsf{TPInit}(\mathsf{prm}, \tau)$, and for any (possibly unbounded) adversary $(\tau', x, \pi) \leftarrow \mathcal{A}(\mathsf{crs})$ we have that if $\tau' \in \mathcal{T}'$ and $\mathsf{V}(\tau_{\mathrm{snd}}, x, \pi) = 1$ then $\exists w : (x, w) \in \mathcal{R}$.

Definition 2 (All-but-One NIZK). *We say that a tag-based NIZK* NIZK *for a relation* $\mathcal{R}$ *and with tag-space* $\mathcal{T}$ *is:*

– *All-but-one Perfect Sound if for all* $\tau \in \mathcal{L}$ *it is* $\{\tau\}$*-tag Composable Perfect Zero-Knowledge and* $\mathcal{T} \setminus \{\tau\}$*-tag Adaptive Perfect Sound.*
– *All-but-one Perfect Hiding and g-Extractable if for all* $\tau \in \mathcal{L}$ *it is* $\mathcal{T} \setminus \{\tau\}$*-tag Composable Perfect Zero-Knowledge and* $\{\tau\}$*-tag Adaptive Perfect g-Extractable.*

For ABO-NIZK we sometimes use the alias ABOInit for the algorithm TPInit.

Construction of an ABO Perfect Hiding. Consider the instantiation of GS Proof system of [EHK+13] based on $\mathcal{D}_k$-MDDH. The common reference string is of the following two forms:

$$[\mathbf{A} \| \mathbf{A}\mathbf{w}] \qquad \text{Perfect Sound Mode}$$
$$[\mathbf{A} \| \mathbf{A}\mathbf{w} - \mathbf{z}] \qquad \text{Perfect Hiding Mode}$$

where $\mathbf{A} \leftarrow\!\!\$\ \mathcal{D}_k$, $\mathbf{w} \leftarrow\!\!\$\ \mathbb{Z}_q^k$ and $\mathbf{z} \notin \mathit{span}(\mathbf{A})$ is a fixed and public vector. We can consider a NIZK with tags where the common reference string is made by two independent CRSs $\mathsf{crs}_1, \mathsf{crs}_2$, both the verifier and the prover on input a tag $\tau \in \mathbb{Z}_q$ derive a CRS $\mathsf{crs}_\tau = \mathsf{crs}_1 + \mathsf{crs}_2 \cdot \tau$. We are ready to define the $\mathsf{ABOInit}_{\mathrm{hid}}$.

$\underline{\mathsf{ABOInit}_{\mathrm{hid}}(\mathsf{prm}, \tau^*)}$:
1. Sample $\mathbf{A}_1, \mathbf{A}_2$ and $\mathbf{w}_1, \mathbf{w}_2$ and set $\mathsf{crs}'_1 = (\mathbf{A}_1 \| \mathbf{A}_1\mathbf{w}_1)$ and $\mathsf{crs}'_2 = (\mathbf{A}_2 \| \mathbf{w}_2 - \mathbf{z})$;
2. Set $\mathsf{crs}_1 = \mathsf{crs}'_1 - \mathsf{crs}'_2 \cdot \tau^*$ and $\mathsf{crs}_2 = \mathsf{crs}'_2$;
3. Output $\mathsf{crs}_1, \mathsf{crs}_2$.

The all-but-one composable zero-knowledge comes readily from the $\mathcal{D}_k$-MDDH assumption and the composable zero-knowledge of GS proofs. The all-but-one adaptive perfect soundness comes readily from the adaptive perfect soundness of GS proofs, in fact we notice that $\mathsf{crs}_{\tau^*} = \mathsf{crs}'_1 - \tau^*\mathsf{crs}'_2 + \tau^*\mathsf{crs}'_2 = \mathsf{crs}'_1$ which allows for perfectly sound proofs.

2.3 The Universal Composability model

In this section, we briefly review some basic notions of the Universal Composability framework introduced by Canetti [Can01]. Let consider the following executions:

$\mathsf{Real}_{\Pi,\mathcal{A},\mathsf{Z}}(\lambda)$**:** run an interaction involving adversary $\mathcal{A}$ and environment Z. When Z generates an input for an honest party, the honest party runs the protocol Π, and gives its output to Z. Finally, Z outputs a value which is taken as the output of $\mathsf{Real}_{\Pi,\mathcal{A},\mathsf{Z}}(\lambda)$.

$\mathsf{Ideal}_{\mathcal{F},\mathsf{S},\mathsf{Z}}(\lambda)$**:** run an interaction involving the simulator S and the environment Z. When Z generates the input for an honest party, the input is passed directly to functionality $\mathcal{F}$, and the corresponding output is given to Z on behalf of that honest party. The output value of Z is taken as the output of $\mathsf{Ideal}_{\mathcal{F},\mathsf{S},\mathsf{Z}}(\lambda)$.

The environment Z provides the inputs to all the parties of the protocols and schedules the order of the messages in the networks. Also, Z decides which party to corrupt: since we consider static corruption, the set of corrupted parties is decided by Z before the protocol starts. Without loss of generality, the environment's final output can be just a single bit: this can be interpreted as the environment's "guess" of whether it is instantiated in the real or ideal world.

Often, for the sake of modularity, it is useful to design a protocol Π that realizes a functionality $\mathcal{F}$ that makes use of other ideal functionalities, e.g., the functionality $\mathcal{G}$. This means that we can define the $\mathcal{G}$-hybrid world, where the parties of Π also interact with the additional functionality $\mathcal{G}$.

$\mathcal{G}$**-**$\mathsf{Hybrid}_{\Pi,\mathcal{A},\mathsf{Z}}(\lambda)$**:** run an interaction involving adversary $\mathcal{A}$, environment Z and the ideal functionality $\mathcal{G}$. When Z generates an input for an honest party, the honest party runs the protocol Π, and gives its output to Z. Finally, Z outputs a value.

The ideal world, instead, interacts only with the ideal functionality $\mathcal{F}$: thus, it is the simulator that simulates the functionality $\mathcal{G}$ to the environment.

Security is captured by the fact that no PPT environment Z can distinguish an execution of the protocol Π (which can interact with the setup assumption $\mathcal{G}$) from a joint execution of the simulator S with the ideal functionality $\mathcal{F}$; this leads to the following definition.

Definition 3. *A protocol Π UC-realizes an ideal functionality $\mathcal{F}$ with setup assumption $\mathcal{G}$ if for all real-world adversaries $\mathcal{A}$ there exists a PPT simulator S such that, for all environments Z:*

$$|\Pr[\mathcal{G}\text{-}\mathsf{Hybrid}_{\Pi,\mathcal{A},\mathsf{Z}}(\lambda) = 1] - \Pr[\mathsf{Ideal}_{\mathcal{F},\mathsf{S},\mathsf{Z}}(\lambda) = 1]| \in \mathsf{negl}(\lambda)$$

Let Π be a protocol that securely realizes an ideal functionality $\mathcal{F}$ in the $\mathcal{G}$-hybrid world, and let Σ be a protocol that securely realizes the ideal functionality $\mathcal{G}$. Then, *composing* Π and Σ, i.e., replacing every invocation of $\mathcal{G}$ with a suitable invocation of Σ, results in a secure protocol for $\mathcal{F}$.

When specifying an ideal functionality, we use the "delayed outputs" terminology adopted in [Can01]: when a functionality $\mathcal{F}$ sends a public delayed output y to a party $\mathcal{P}$, we mean that y is first sent to the simulator S and then forwarded to $\mathcal{P}$ only after an acknowledgment by S.

Experiment $\mathbf{Exp}^{\mathtt{lRCCA}}_{\mathcal{A},\mathsf{PKE},f}(\lambda, b)$

$\mathsf{prm} \leftarrow \mathsf{Setup}(1^\lambda)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{prm})$
$(\mathtt{M}_0, \mathtt{M}_1, z) \leftarrow \mathcal{A}_1^{\mathsf{ODec}}(\mathsf{pk})$
$\mathtt{C} \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{pk}, \mathtt{M}_b)$
$z' \leftarrow \mathcal{A}_2^{\mathsf{ODec}}(\mathtt{C}, z)$
$b' \leftarrow \mathcal{A}_3(f(\mathsf{sk}), z')$
return $b' \stackrel{?}{=} b$

Oracle $\mathsf{ODec}(\mathtt{C})$

$\mathtt{M} \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathtt{C})$
if $\mathtt{M} \in \{\mathtt{M}_0, \mathtt{M}_1\}$:
  return $\diamond$
return $\mathtt{M}$

Fig. 1. The lRCCA security experiment.

For the sake of simplicity, we assume that all the ideal functionalities have an implicit public parameter prm hardcoded. We can think of prm as being the description of a cryptographic group or some other publicly-available information. We tweak the definitions of both the hybrid and the ideal world to include such public parameters; specifically, we consider that the ideal world (resp. hybrid world) samples $\mathsf{prm} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$ and passes along this information, together with the security parameters, to all the ITMs involved in the execution of the protocol.

3 Definitions

3.1 Replayable CCA with Leakage Security

We rely on the following notion of security for Rand-PKE.

Definition 4 (RCCA with leakage Security). *Consider the experiment* $\mathbf{Exp}^{\mathtt{lRCCA}}_{\mathcal{A},\mathsf{PKE},f}$ *in Fig. 1, with parameters* λ*, an adversary* $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$*, a PKE scheme* PKE*, and a leakage function* f*. We say that* PKE *is* leakage-resilient replayable CCA-secure (lRCCA-secure) *w.r.t. a leakage function* f *if for any PPT adversary* $\mathcal{A}$*:*

$$\mathbf{Adv}^{\mathtt{lRCCA}}_{\mathcal{A},\mathsf{PKE},f}(\lambda) := \left|2\Pr\left[\mathbf{Exp}^{\mathtt{lRCCA}}_{\mathcal{A},\mathsf{PKE},f}(\lambda, b) = 1, b \leftarrow\!\!\$\ \{0,1\}\right] - 1\right| \in \mathsf{negl}(\lambda).$$

We note that the above experiment is identical to a classical RCCA security game, with the difference that here $\mathcal{A}$ is given the additional leak information $f(\mathsf{sk})$ just before committing to the verdict bit b'. $\mathcal{A}$ cannot invoke the decryption oracle after the leak occurs.

3.2 The Verify-then-Decrypt Ideal Functionality

We give in Fig. 2 the formal definition of this ideal functionality. Informally, the ideal functionality accepts two lists of ciphertexts, such that the first list includes

Functionality $\mathcal{F}_{\mathsf{VtDec}}^{\mathsf{PKE},f}$

The ideal functionality has as parameters a public-key encryption scheme $\mathsf{PKE} := (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$, an efficiently-computable function f and (implicit) group parameters $\mathsf{prm} \in \mathsf{Setup}(1^\lambda)$. The functionality interacts with m parties $\mathcal{P}_i$ and with an adversary S.

Public Key. Upon message $(\mathsf{KEY}, \mathsf{sid})$ from a party $\mathcal{P}_i$, $i \in [m]$, if $(\mathsf{sid}, \mathsf{pk}, \mathsf{sk})$ is not in the database sample $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{prm})$ and store the tuple $(\mathsf{sid}, \mathsf{pk}, \mathsf{sk})$ in the database. Send $(\mathsf{KEY}, \mathsf{sid}, \mathsf{pk})$ to $\mathcal{P}_i$.

Verify then Decrypt. Upon message $(\mathsf{VTDEC}, \mathsf{sid}, C_V, C_D)$ from party $\mathcal{P}_i$:

- If the tuple $(\mathsf{sid}, \mathsf{pk}, \mathsf{sk})$ does not exist in the database, ignore the message.
- Check that a tuple $(\mathsf{sid}, C_V, C_D, \mathcal{I})$ where $\mathcal{I} \subseteq [m]$ exists in the database; if so, update $\mathcal{I}$ including the index i, otherwise create the new entry $(\mathsf{sid}, C_V, C_D, \{i\})$ in the database.

If $|\mathcal{I}| = m$ and $C_D \subseteq C_V$ then:

- Send $(\mathsf{sid}, f(\mathsf{sk}))$ to the adversary S.
- Parse C_V as $(\mathsf{C}_i^V)_{i \in [|C_V|]}$ and C_D as $(\mathsf{C}_i^D)_{i \in [|C_D|]}$
- Compute the vector $\mathbf{b} \in \{0,1\}^{|C_V|}$ such that for any i, $b_i = 1$ iff $\mathsf{Dec}(\mathsf{sk}, \mathsf{C}_i^V) \neq \perp$.
- If $\exists i : b_i = 0$ set $M_o := ()$, else compute $M_o := (\mathsf{Dec}(\mathsf{sk}, \mathsf{C}_i^D))_{i \in [|C_D|]}$, send a public delayed output $(\mathsf{VTDEC}, \mathsf{sid}, \mathbf{b}, M_o)$ to the parties $\mathcal{P}_i$ for $i \in [m]$,

Fig. 2. UC ideal functionality for Verify-then-Decrypt.

Functionality $\mathcal{F}_{\text{Mix}}$

The functionality has n sender parties $\mathcal{P}_{S_i}$, m mixer parties $\mathcal{P}_{M_i}$.

Input. Upon activation on message $(\mathsf{INPUT}, \mathsf{sid}, \mathtt{M})$ from $\mathcal{P}_{S_i}$ (or the adversary if $\mathcal{P}_{S_i}$ is corrupted), if $i \in L_{S,\mathsf{sid}}$ ignore the message else register the index i in the list of the senders $L_{S,\mathsf{sid}}$ and register the message $\mathtt{M}$ in the list $L_{I,\mathsf{sid}}$ of the input messages. Notify the adversary that the sender $\mathcal{P}_{S_i}$ has sent an input.

Mix. Upon activation on message $(\mathsf{MIX}, \mathsf{sid})$ from $\mathcal{P}_{M_i}$ (or the adversary if $\mathcal{P}_{M_i}$ is corrupted), register the index i in the list of the mixers $L_{\text{mix},\mathsf{sid}}$ and notify the adversary.

Delivery. Upon activation on message $(\mathsf{DELIVER}, \mathsf{sid})$ from the adversary $\mathcal{S}$ If $|L_{\text{mix},\mathsf{sid}}| = m$ and $|L_{S,\mathsf{sid}}| = n$ then send a public delayed output $M_{\mathsf{sid}} \leftarrow \mathsf{Sort}(L_{I,\mathsf{sid}})$ to all the mixer parties.

Fig. 3. UC ideal functionality for MixNet.

all the ciphertexts in the second list, it first verifies that all the ciphertexts in the first list decrypt to valid messages (i.e. no decryption error) and releases such output together with the decryption from the second list. The functionality has parameter f that denotes the leakage of secret information allowed to realize such functionality.

4 Mix-Net

We now describe our mixnet protocol that UC-realizes the ideal functionality $\mathcal{F}_{\text{Mix}}$ with setup assumptions $\mathcal{F}_{\mathsf{VtDec}}$ and $\mathcal{F}_{\mathsf{crs}}$. We start by giving the definition of Sumcheck-Admissible relation with respect to a PKE. In this definition we abstract the necessary property for the zero-knowledge proof system used by the mixers in the protocol.

Definition 5 (Sumcheck-Admissible Relation w.r.t. PKE). *Let PKE be a public-key encryption scheme with public space $\mathcal{PK}$ and the ciphertext space being a subset of $\mathcal{CT}$. For any λ, any $\mathsf{prm} \in \mathsf{Setup}(1^\lambda)$, let $\mathcal{R}_{ck}^{\mathsf{prm}} : (\mathcal{PK} \times \mathcal{CT}^{2n}) \times \{0,1\}^*$ be an NP-relation. We parse an instance of $\mathcal{R}_{ck}^{\mathsf{prm}}$ as $x = (\mathsf{pk}, L_1, L_2)$ where $L_j = (\mathtt{C}_i^j)_{i\in[n]}$ for $j \in \{1,2\}$. $\mathcal{R}_{ck}$ is Sumcheck-Admissible w.r.t. PKE if:*

(Sumcheck) *For any $(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{KGen}(\mathsf{prm})$ and for any $x \coloneqq (\mathsf{pk}, L_1, L_2)$ we have that if $x \in \mathcal{L}(\mathcal{R}_{ck})$ and $\forall j, i : \mathsf{Dec}(\mathsf{sk}, \mathtt{C}_i^j) \neq \bot$ then $\sum_i \mathsf{Dec}(\mathsf{sk}, \mathtt{C}_i^1) - \mathsf{Dec}(\mathsf{sk}, \mathtt{C}_i^2) = 0$.*

(Re-Randomization Witness) *For any $(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{KGen}(\mathsf{prm})$ and for any $x \coloneqq (\mathsf{pk}, L_1, L_2)$ such that there exists $(r_i)_{i\in[n]}$ where $\forall i \in [n], \exists j \in [n] : \mathtt{C}_i^2 = \mathsf{Rand}(\mathsf{pk}, \mathtt{C}_j^1; r_i)$ we have that $(x, (r_i)_{i\in[n]}) \in \mathcal{R}_{ck}$.*

Building Blocks. Let PKE be a Rand-PKE scheme, let f be any efficiently-

Functionality $\mathcal{F}_{\mathrm{CRS}}^{\mathsf{Init}}$

The functionality interacts with n parties $\mathcal{P}_i$ and an adversary S and has parameters a PPT algorithm Init that outputs obliviously sampleable common-reference string and an (implicit) public parameter prm.

Initialization. Upon activation, sample $\mathsf{crs} \leftarrow\$ \mathsf{Init}(\mathsf{prm})$ and store it.
Public Value. Upon activation on message CRS from a party $\mathcal{P}_i$, $i \in [n]$,send crs to $\mathcal{P}_i$.

Fig. 4. UC ideal functionality for Common Reference String, parametrized by group parameters prm and a NIZK setup Init.

computable function and let $\mathcal{R}_{\mathrm{ck}}$ be any Sumcheck-Admissible relation w.r.t. PKE. The building blocks for our Mix-Net are:

1. A Rand-PKE scheme PKE that is lRCCA-secure w.r.t. f according to Definition 4.
2. An All-but-One Perfect-Sound tag-based NIZK (cfr. Section 2.2) $\mathsf{NIZK}_{\mathsf{mx}} := (\mathsf{Init}_{\mathsf{mx}}, \mathsf{P}_{\mathsf{mx}}, \mathsf{V}_{\mathsf{mx}})$ for proving membership in the relation $\mathcal{R}_{\mathrm{ck}}$, with tag space $[m]$.
3. An All-but-One Perfect-Hiding tag-based NIZK $\mathsf{NIZK}_{\mathsf{sd}} = (\mathsf{Init}_{\mathsf{sd}}, \mathsf{P}_{\mathsf{sd}}, \mathsf{V}_{\mathsf{sd}})$ for knowledge of the plaintext, i.e. a NIZK for the relation $\mathcal{R}_{\mathrm{msg}} := \{(\mathsf{pk}, \mathtt{C}), (\mathtt{M}, r) : \mathtt{C} = \mathsf{Enc}(\mathsf{pk}, \mathtt{M}; r)\}$, with tag space $[n]$. In particular, a weaker notion of extractability that guarantees that the message $\mathtt{M}$ is extracted is sufficient.
4. An ideal functionality $\mathcal{F}_{\mathsf{VtDec}}^{\mathsf{PKE},f}$, as defined in Fig. 2.
5. An ideal functionality for the common reference string (see Fig. 4) of the above NIZKs. In particular, the functionality initializes a CRS $\mathsf{crs}_{\mathsf{mx}}$ for $\mathsf{NIZK}_{\mathsf{mx}}$, and an additional CRS $\mathsf{crs}_{\mathsf{sd}}$ for $\mathsf{NIZK}_{\mathsf{sd}}$.

Finally, we implicitly assume that all parties have access to point-to-point authenticated channels.

Protocol Description. To simplify the exposition, we describe in this section the case of a single invocation, i.e. the protocol is run only once with a single, fixed session identifier sid; in Fig. 5 we describe in detail the protocol for the general case of a multi-session execution. At the first activation of the protocol, both the mixer parties and the sender parties receive from the functionality $\mathcal{F}_{\mathsf{VtDec}}$ the public key pk for the scheme PKE and the CRSs from $\mathcal{F}_{\mathrm{CRS}}$. At submission phase, each sender $\mathcal{P}_{S_i}$ encrypts their input message $\mathtt{M}_i$ by computing $\mathtt{C}_i \leftarrow\$ \mathsf{Enc}(\mathsf{pk}, \mathtt{M}_i)$, and attaches a NIZK proof of knowledge π_i^{sd} of the plaintext, using i as tag. Finally, the party $\mathcal{P}_{S_i}$ broadcasts their message $(\mathtt{C}_i, \pi_{\mathsf{sd}}^i)$. After all sender parties have produced their ciphertexts, the mixers, one by one, shuffle their input lists and forward to the next mixer their output lists. In particular, the party $\mathcal{P}_{M_i}$ produces a random permutation of the input list of ciphertexts L_{i-1} (L_0 is the list of ciphertexts from the senders) by re-randomizing each

Protocol Π_{Mix}

Input. Upon activation on message $(\mathsf{INPUT}, \mathsf{sid}, \mathtt{M})$, $\mathcal{P}_{S_i}$ computes $\mathtt{C} \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{pk}, \mathtt{M})$, and $\pi_{\mathsf{sd}} \leftarrow\!\!\$\ \mathsf{P}_{\mathsf{sd}}(\mathsf{crs}_{\mathsf{sd}}, i, (\mathsf{pk}, \mathtt{C}), (\mathtt{M}, r))$. Broadcasts $(\mathsf{sid}, i, \mathtt{C}, \pi_{\mathsf{sd}})$.

Mix. Upon activation, the party $\mathcal{P}_{M_i}$, depending on its state, does as follow:

- If it is the first activation with message $(\mathsf{MIX}, \mathsf{sid})$ from the environment sends the message $(\mathsf{KEY}, \mathsf{sid})$ to $\mathcal{F}_{\mathsf{VtDec}}$ and return.
- If the message $(\mathsf{KEY}, \mathsf{sid}, \mathsf{pk})$, the messages $(\mathsf{sid}, i, \mathtt{C}, \pi_{\mathsf{sd}})$ for all the senders and the messages $(\mathsf{sid}, L_j, \pi^j_{\mathsf{mx}})$ for all the mixers with index $j \leq i-1$ were received:
  1. Samples a permutation ζ_i
  2. Reads the pair message $(L_{i-1}, \pi^i_{\mathsf{mx}})$ sent by the party $\mathcal{P}_{M_{i-1}}$ (or simply reads L_0 if this is the first mixer party)
  3. Shuffles and re-randomizes the list of ciphertexts: produces the new list $L_i = (\mathtt{C}'_{\zeta_i(j)})_{j\in[n]}$ where $\mathtt{C}'_j \leftarrow \mathsf{Rand}(\mathsf{pk}, \mathtt{C}_{i-1}; r_j)$ and r_j uniformly random string.
  4. Computes the sumcheck proof for the two lists of ciphertexts $\pi^i_{\mathsf{mx}} \leftarrow\!\!\$\ \mathsf{P}_{\mathsf{mx}}(\mathsf{crs}_{\mathsf{mx}}, (\mathsf{pk}, L_1, L_2), (r_j)_{j\in[n]})$
  5. Sends to all the mixers $(\mathsf{sid}, L_i, \pi^i_{\mathsf{mx}})$.
- If the message $(\mathsf{sid}, L_m, \pi^m_{\mathsf{mx}})$ was received, checks that all the mixer proofs π^i_{mx}, for $i \in [m]$ accept, else abort.
- Computes $L \coloneqq \mathsf{Concat}(L_1, \dots, L_m)$ and sends $(\mathsf{VtDEC}, \mathsf{sid}, L, L_m)$ to $\mathcal{F}_{\mathsf{VtDec}}$
- If the message $(\mathsf{sid}, \mathbf{b}, M_o)$ from $\mathcal{F}_{\mathsf{VtDec}}$ was received, if $\exists i : b_i = 0$ then returns $\perp$, else computes and returns $L_o \coloneqq \mathsf{Sort}(M_o)$

Fig. 5. Our protocol Π_{Mix}.

ciphertext in the list and then permuting the whole list, thus computing a new list L_i. Additionally, the mixer computes a NIZK proof of membership π^i_{mx} with tag i, for the instance $(\mathsf{pk}, L_{i-1}, L_i)$ being in the sumcheck-admissible relation, because of the re-randomization witness property of Definition 5, the mixer holds a valid witness for such an instance. After this phase, the mixers are ready for the verification: the mixers invoke the Verify-then-Decrypt functionality $\mathcal{F}_{\mathsf{VtDec}}$ to (i) verify that each list seen so far is made up only of valid ciphertexts and (ii) decrypt the ciphertexts contained in the final list. Finally publishes the list of the messages received by $\mathcal{F}_{\mathsf{VtDec}}$, sorted according to some common deterministic criterion, e.g. the lexicographical order.

Theorem 1. *For any arbitrary leakage function f, if PKE is lRCCA-secure w.r.t. f, $\mathsf{NIZK}_{\mathsf{mx}}$ is ABO Perfect Sound, $\mathsf{NIZK}_{\mathsf{sd}}$ is ABO Perfect Hiding, then the protocol described in Fig. 5 UC-realizes the functionality $\mathcal{F}_{\text{Mix}}$, described in Fig. 3, with setup assumptions $\mathcal{F}^{\mathsf{PKE},f}_{\mathsf{VtDec}}$ and $\mathcal{F}_{\mathsf{crs}}$.*

Proof. We now prove the existence of a simulator S, and we show that no PPT environment Z can distinguish an interaction with the real protocol from an interaction with S and the ideal functionality $\mathcal{F}_{\mathrm{Mix}}$ (the ideal world), i.e. the distribution $(\mathcal{F}_{\mathsf{VtDec}}, \mathcal{F}_{\mathsf{crs}})\text{-}\mathsf{Hybrid}_{Z,\Pi_{\mathrm{Mix}},\mathcal{A}}(\lambda)$ is indistinguishable from $\mathsf{Ideal}_{Z,\mathcal{F}_{\mathrm{Mix}},\mathsf{S}}(\lambda)$. In our proof, we give a sequence of hybrid experiments in which the $(\mathcal{F}_{\mathsf{VtDec}}, \mathcal{F}_{\mathsf{crs}})$-hybrid world is progressively modified until reaching an experiment that is identically distributed to the ideal world. In what follows, we indicate with h^* the index of the first honest mixer. For label $\in \{\mathrm{in}, \mathrm{hide}\}$, we introduce the set Ψ_{label} consisting of tuples (x, y). We define the functions ψ_{label} and $\psi^{-1}_{\mathrm{label}}$ associated with the corresponding set:

$$\psi_{\mathrm{label}}(x) \coloneqq \begin{cases} y & \text{if } (x,y) \in \Psi_{\mathrm{label}} \\ x & \text{otherwise} \end{cases} \qquad \psi^{-1}_{\mathrm{label}}(y) \coloneqq \begin{cases} x & \text{if } (x,y) \in \Psi_{\mathrm{label}} \\ y & \text{otherwise} \end{cases}$$

Informally, the pair of functions $\psi_{\mathrm{in}}, \psi^{-1}_{\mathrm{in}}$ helps the hybrids to keep track of the ciphertexts sent by the honest senders while they are mixed by the first $h^* - 1$ mixers, while the pair of functions $\psi_{\mathrm{hide}}, \psi^{-1}_{\mathrm{hide}}$ helps to keep track of the ciphertexts output by the first honest mixer while they are mixed by the remaining mixers in the chain. We recall that in the protocol the mixers $\mathcal{P}_{M_i}$, for $i \in [m]$, send a message which includes a list L_i of ciphertexts. Whenever it is convenient we parse L_i as $(\mathtt{C}_{i,j})_{j\in[n]}$. Let `Invalid` be the event that, during the interaction of Z with the simulator/protocol, there exist $i \in [m], j \in [n]$ such that $\mathsf{Dec}(\mathsf{sk}, \mathtt{C}_{i,j}) = \bot$ or $\mathsf{Vf}(\mathsf{crs}_{\mathsf{mx}}, (\mathsf{pk}, L_{i-1}, L_i), \pi^i_{\mathsf{mx}}) = 0$ (namely, π^i_{mx} does not verify). Clearly, when the event `Invalid` occurs, the protocol aborts.

Hybrid $\mathbf{H}_0$. This first hybrid is equivalent to $(\mathcal{F}_{\mathsf{VtDec}}, \mathcal{F}_{\mathsf{crs}})\text{-}\mathsf{Hybrid}_{Z,\Pi_{\mathrm{Mix}},\mathcal{A}}(\lambda)$.

Hybrid $\mathbf{H}_1$. In this hybrid, we change the way $\mathsf{crs}_{\mathsf{mx}}$ is generated. We run $(\mathsf{crs}_{\mathsf{mx}}, \mathsf{tps}) \leftarrow\$ \mathsf{ABOInit}(\mathsf{prm}, h^*)$. Also, the proof $\pi^{h^*}_{\mathsf{mx}}$ of the first honest mixer is simulated. This hybrid is indistinguishable from the previous one because of the ABO Composable Perfect Zero-Knowledge property of the NIZK (cfr. Section 2.2).

Hybrid $\mathbf{H}_2$. The first honest mixer $\mathcal{P}_{M_{h^*}}$, rather than re-randomizing the ciphertexts received in input, decrypts and re-encrypts all the ciphertexts. If the decryption fails for some ciphertext $\mathtt{C}_i$, $\mathcal{P}_{M_{h^*}}$ re-randomizes this "invalid" ciphertext and continues. This hybrid is indistinguishable from the previous one because the PKE scheme PKE is perfectly re-randomizable (cfr. Definition 1): because of the tightness of the decryption property, we have that $\forall j$, if $\mathsf{Dec}(\mathsf{sk}, \mathtt{C}_{h^*-1,j}) = \mathtt{M}_{h^*-1,j} \neq \bot$ then $\mathtt{C}_{h^*,j} \in \mathsf{Enc}(\mathsf{pk}, \mathtt{M}_{h^*-1,j})$ with overwhelming probability; also, by the indistinguishability property, the distribution of the re-randomized ciphertext $\mathsf{Rand}(\mathsf{pk}, \mathtt{C}_{h^*-1,j})$ and a fresh encryption $\mathsf{Enc}(\mathsf{pk}, \mathtt{M}_{h^*-1,j})$ are statistically close.

Hybrid $\mathbf{H}_3$. Here we introduce the set Ψ_{hide} and we populate it with the pairs $(\mathtt{M}_{h^*-1,i}, \mathtt{H}_i)_{i\in[n]}$, where the messages $\mathtt{H}_1, \ldots, \mathtt{H}_n$ are distinct and sampled at random from the message space $\mathcal{M}$. When we simulate the ideal functionality $\mathcal{F}_{\mathsf{VtDec}}$, we output $\psi^{-1}_{\mathrm{hide}}(\mathtt{M})$ for all successfully decrypted messages $\mathtt{M}$. The only event that can distinguish the two hybrids is the event that $\neg$`Invalid` and $\exists j, j' : \mathsf{Dec}(\mathsf{sk}, \mathtt{C}_{m,j}) = \mathtt{H}_{j'}$. However, the messages $\mathtt{H}_1, \ldots, \mathtt{H}_n$ are not in the view of Z, thus the probability of such event is at most $\frac{n^2}{|\mathcal{M}|}$. This hybrid and the previous one are statistically indistinguishable.

Hybrid $\mathbf{H}_4$. In this hybrid, rather than re-encrypting the same messages, the first honest mixer re-encrypts the fresh and uncorrelated messages $\mathtt{H}_1, \ldots, \mathtt{H}_n$ (used to populate Ψ_{hide}). Specifically, $\mathcal{P}_{M_{h^*}}$ samples a random permutation ζ_{h^*} and computes the list $L_{h^*} := (\mathtt{C}_{h^*,j})_{j\in[n]}$, with $\mathtt{C}_{h^*,\zeta_{h^*}(j)} \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{pk}, \psi_{\text{hide}}(\mathtt{M}_{h^*-1,j}))$. This hybrid is indistinguishable from the previous one, and the proof can be reduced to the lRCCA security of the scheme PKE.

Lemma 1. *Hybrids $\boldsymbol{H}_3$ and $\boldsymbol{H}_4$ are computationally indistinguishable.*

Proof. We use a hybrid argument. Let $\mathbf{H}_{3,i}$ be the hybrid game in which the first honest mixer computes the list $L_{h^*} := (\mathtt{C}_{h^*,j})_{j\in[n]}$ as:

$$\mathtt{C}_{h^*,\zeta_{h^*}(j)} := \begin{cases} \mathsf{Enc}(\mathsf{pk}, \psi_{\text{hide}}(\mathtt{M}_{h^*-1,j})) & \text{if } j \le i \\ \mathsf{Enc}(\mathsf{pk}, \mathtt{M}_{h^*-1,j}) & \text{if } j > i \end{cases}$$

In particular, it holds that $\mathbf{H}_3 \equiv \mathbf{H}_{3,0}$ and $\mathbf{H}_4 \equiv \mathbf{H}_{3,n}$. We prove that $\forall i \in [n]$ the hybrid $\mathbf{H}_{3,i-1}$ is computationally indistinguishable from $\mathbf{H}_{3,i}$, reducing to the lRCCA-security of the scheme PKE. Consider the following adversary against the lRCCA-security experiment.

Adversary $\mathcal{B}(\mathsf{pk})$ with oracle access to $\mathsf{ODec}(\cdot)$.

- Simulate the hybrid experiment $\mathbf{H}_{3,i-1}$, in particular, when the environment instructs a corrupted mixer to send the message $(\mathsf{KEY}, \mathsf{sid})$ simulate the ideal functionality $\mathcal{F}_{\mathsf{VtDec}}$ sending back the answer $(\mathsf{KEY}, \mathsf{sid}, \mathsf{pk})$.
- When it is time to compute the list of the first honest mixer L_{h^*}, namely, when the mixer $\mathcal{P}_{M_{h^*}}$ is activated by the environment and has received for all $j \in [n]$ the messages $(\mathsf{sid}, j, \mathtt{C}, \pi_{\mathsf{sd}})$ from the senders and the messages $(\mathsf{sid}, L_j, \pi^j_{\mathsf{mx}})$ from all the mixers with index $j \le h^* - 1$, first decrypt all the ciphertexts received so far using oracle access to $\mathsf{ODec}(\cdot)$. Let $\mathtt{M}_{h^*-1,i}$ be the decryption of the ciphertext $\mathtt{C}_{h^*-1,i}$. If $\mathtt{M}_{h^*-1,i} = \perp$ then output a random bit, else send the pair of messages $(\mathtt{M}_{h^*-1,i}, \mathtt{H}_i)$ to the lRCCA challenger, thus receiving a challenge ciphertext $\mathtt{C}^*$.
- Populate the list L_{h^*} by setting $\mathtt{C}_{\zeta_{h^*}(i)} \leftarrow \mathtt{C}^*$, and computing all the other ciphertexts as described in $\mathbf{H}_{3,i-1}$. Continue the simulation as the hybrid does.
- When all the mixers have sent the message (VtDEC, L, L_m), to $\mathcal{F}_{\mathsf{VtDec}}$, check that all the mixer proofs accept, otherwise abort the simulation and output a random bit. Then decrypt all the ciphertexts in L by sending queries to the guarded decryption oracle, i,e. send the query $\mathtt{C}_{i',j}$, receiving back the message $\mathtt{M}_{i',j} \in \mathcal{M} \cup \{\diamond, \perp\}$. If $\mathtt{M}_{i',j} = \perp$, abort and output a random bit. If $\mathtt{M}_{i',j} = \diamond$, then set $\mathtt{M}_{i',j} := \mathtt{M}_{h^*-1,i}$. Simulate the leakage from $\mathcal{F}_{\mathsf{VtDec}}$ through the leakage received by the lRCCA security experiment: in particular, the reduction loses access to the guarded decryption oracle, receives the value $f(\mathsf{sk})$ and sends the message $(\mathsf{sid}, \mathbf{b}, \{\mathtt{M}_{m,j}\}_{j\in[n]})$ as required by the protocol.

– Finally, forward the bit returned by Z.

First we notice that when the guarded decryption oracle returns a message $\mathtt{M}_{i',j} = \diamond$ then the reduction can safely return $\mathtt{M}_{h^*-1,i}$. In fact, the ciphertext would decrypt to either $\mathtt{H}_i$ or to $\mathtt{M}_{h^*-1,i}$, however by the change introduced in $\mathbf{H}_3$, we have that $\mathtt{M}_{h^*-1,i} = \psi^{-1}_{\text{hide}}(\mathtt{H}_i)$ and $\mathtt{M}_{h^*-1,i} = \psi^{-1}_{\text{hide}}(\mathtt{M}_{h^*-1,i})$.

It is easy to see that when the challenge bit b of the experiment is equal to 0, the view of Z is identically distributed to the view in $\mathbf{H}_{3,j-1}$, while if the challenge bit is 1, the view of Z is identically distributed to the one in $\mathbf{H}_{3,j}$. Thus $|\Pr[\mathbf{H}_{3,j-1}(\lambda) = 1] - \Pr[\mathbf{H}_{3,j}(\lambda) = 1]| \leq \mathbf{Adv}^{\mathtt{lRCCA}}_{\mathcal{B},\mathsf{PKE},f}(\lambda)$.

Hybrid $\mathbf{H}_5$. Let $V_m \coloneqq (\mathtt{M}_{m,j})_{j\in[n]}$ (resp. $V_{h^*} \coloneqq (\mathtt{M}_{h^*,j})_{j\in[n]}$) be the list of decrypted ciphertexts output by the last mixer $\mathcal{P}_{M_m}$ (resp. by the first honest mixer $\mathcal{P}_{M_{h^*}}$). In the hybrid $\mathbf{H}_5$ the simulation aborts if $\neg\mathtt{Invalid}$ and $V_m \neq V_{h^*}$.

Lemma 2. *Hybrids $\boldsymbol{H}_4$ and $\boldsymbol{H}_5$ are computationally indistinguishable.*

Proof. Since $|V_m| = |V_{h^*}|$ and the messages $\mathtt{H}_1, \ldots, \mathtt{H}_n$ are distinct, the event $V_{h^*} \neq V_m$ holds if and only if there exists an index $j \in [n]$ such that $\mathsf{Count}(\mathtt{H}_j, V_m) \neq 1$. Let $\mathbf{H}_{4,i}$ be the same as $\mathbf{H}_4$ but the simulation aborts if $\neg\mathtt{Invalid}$ and $\exists j \in [i] : \mathsf{Count}(\mathtt{H}_j, V_m) \neq 1$. Clearly, $\mathbf{H}_{4,0} \equiv \mathbf{H}_4$ and $\mathbf{H}_{4,n} \equiv \mathbf{H}_5$. Let $\mathtt{Bad}_i$ be the event that $(\neg\mathtt{Invalid} \wedge \mathsf{Count}(\mathtt{H}_i, V_m) \neq 1)$. It is easy to check that:

$$|\Pr[\mathbf{H}_{4,i-1}(\lambda) = 1] - \Pr[\mathbf{H}_{4,i}(\lambda) = 1]| \leq \Pr[\mathtt{Bad}_i].$$

In fact, the two hybrids are equivalent if the event $\mathtt{Bad}_i$ does not happen.

We define an adversary to the lRCCA security of PKE that makes use of the event above.

Adversary $\mathcal{B}(\mathsf{pk})$ with oracle access to $\mathsf{ODec}(\cdot)$.

1. Simulate the hybrid experiment $\mathbf{H}_5$; in particular, when the environment instructs a corrupted mixer to send the message $(\mathsf{KEY}, \mathsf{sid})$ simulate the ideal functionality $\mathcal{F}_{\mathsf{VtDec}}$ sending back the answer $(\mathsf{KEY}, \mathsf{sid}, \mathsf{pk})$. (Thus embedding the public key from the challenger in the simulation.)
2. When it is time to compute the list of the first honest mixer L_{h^*}, namely, when the mixer $\mathcal{P}_{M_{h^*}}$ is activated by the environment and has received the messages $(\mathsf{sid}, i, \mathtt{C}, \pi_{\mathsf{sd}})$ for all the senders and the messages $(\mathsf{sid}, L_j, \pi^j_{\mathsf{mx}})$ for all the mixers with index $j \leq h^* - 1$, first decrypt all the ciphertexts received so far using oracle access to the guarded decryption oracle. If there is a decryption error, output a random bit b'.
3. Sample $\mathtt{H}^{(0)}, \mathtt{H}^{(1)} \leftarrow\!\!\$\ \mathcal{M}$ and send the pair of messages $(\mathtt{H}^{(0)}, \mathtt{H}^{(1)})$ to the lRCCA challenger, receiving back the challenge ciphertext $\mathtt{C}^*$. Set the list $L_{h^*} = (\mathtt{C}_{h^*,j})_{j\in[n]}$ as follow:
$$\mathtt{C}_{h^*,\zeta_{h^*}(j)} \coloneqq \begin{cases} \mathsf{Enc}(\mathsf{pk}, \mathtt{M}_{h^*-1,j}) & \text{if } j \neq i \\ \mathtt{C}^* & \text{else} \end{cases}$$
where recall that ζ_{h^*} is the random permutation used by the h^*-th mixer. Continue the simulation as the hybrid does.

4. When all the mixer have sent the message (VtDEC, L, L_m), to $\mathcal{F}_{\mathsf{VtDec}}$, decrypt all of the ciphertexts in L by sending queries to the guarded decryption oracle, namely, send the query $\mathtt{C}_{i',j}$ for all $i' > h^*$ and all $j \in [n]$, receiving back as answer the plaintext messages $\mathtt{M}_{i',j} \in \mathcal{M} \cup \{\diamond, \bot\}$.
   If the event $\mathtt{Invalid}$ holds, then abort the simulation and output a random bit b'.
5. Let $C \leftarrow \mathsf{Count}(\diamond, V_m)$, if $C = 1$ then abort the simulation and output a random bit b'.
6. From now one we can assume that $\neg\mathtt{Invalid}$ and $C \neq 1$; Compute

$$\mathtt{M} \leftarrow (C-1)^{-1} \cdot \left(\sum_{j \in [n], \mathtt{M}_{m,j} \neq \diamond} \mathtt{M}_{m,j} - \sum_{j \neq \zeta_{h^*}(i)} \mathtt{M}_{h^*,j} \right). \tag{1}$$

   Output b' s.t. $\mathtt{M} = \mathtt{H}^{(b')}$.

First, we notice that the simulation $\mathcal{B}$ provides to the environment Z is perfect, indeed, independently of the challenge bit, the message $\mathtt{H}^{(b)}$ is distributed identically to $\mathtt{H}_j$. Thus the probability that $\mathtt{Bad}_i$ happens in the reduction is the same as the probability the event happens in the hybrid experiments.

Let $\mathtt{Abort}$ be the event that $\mathcal{B}$ aborts and outputs a random bit. Notice that:

$$\mathtt{Abort} \equiv \mathtt{Invalid} \vee (C = 1).$$

Let $\mathtt{Wrong}$ be the event that $\exists j : \mathsf{Dec}(\mathsf{sk}, \mathtt{C}_{m,j}) = \mathtt{H}^{(1-b)}$; notice that the message $\mathtt{H}^{(1-b)}$ is independent of the view of the environment Z, thus the probability of $\mathtt{Wrong}$ is at most $n/|\mathcal{M}|$. Moreover, we have $\mathtt{Bad}_i \equiv \neg\mathtt{Abort} \wedge \neg\mathtt{Wrong}$ because, by definition of $\neg\mathtt{Wrong}$, all the ciphertexts that decrypt to $\diamond$ in L_m are indeed an encryption of $\mathtt{H}^{(b)}$; thus, assuming the event holds, $C \neq 1$ iff $\mathsf{Count}(\mathtt{H}^{(b)}, V_m) \neq 1$. The probability of guessing the challenge bit when $\mathcal{B}$ aborts is $\frac{1}{2}$, thus we have:

$$\Pr[b = b'] \geq \tfrac{1}{2} \Pr[\neg\mathtt{Bad}_i] + \Pr[b = b' | \mathtt{Bad}_i] \Pr[\mathtt{Bad}_i] - \tfrac{n}{|\mathcal{M}|} \tag{2}$$

We now compute the probability that $b = b'$ conditioned on $\mathtt{Bad}_i$. First notice that $\neg\mathtt{Invalid}$ implies that the ciphertexts in the lists $L_{h^*}, \dots, L_m$ decrypt correctly and that the proofs π^j_{mx} for $j > h^*$ verify. Thus by applying the sumcheck-admissibility w.r.t. PKE of the relation $\mathcal{R}_{\mathsf{mx}}$ and by the ABO perfect soundness of $\mathsf{NIZK}_{\mathsf{mx}}$ we have:

$$\sum_{j \in [n]} \mathsf{Dec}(\mathsf{sk}, \mathtt{C}_{h^*,j}) - \sum_{j \in [n]} \mathsf{Dec}(\mathsf{sk}, \mathtt{C}_{m,j}) = 0.$$

If we condition on $\neg\mathtt{Wrong}$ then:

$$\left(\mathtt{H}^{(b)} + \sum_{j \neq \zeta_{h^*}(j^*)} \mathtt{M}_{h^*,j} \right) - \left(C \cdot \mathtt{H}^{(b)} + \sum_{j \in [n], \mathtt{M}_{m,j} \neq \diamond} \mathtt{M}_{m,j} \right) = 0.$$

By solving the above equation for $\mathtt{H}^{(b)}$, we obtain $\mathtt{M} = \mathtt{H}^{(b)}$, therefore $\mathcal{B}$ guesses the challenge bit with probability 1 when conditioning on $\neg\mathtt{Abort} \wedge \neg\mathtt{Wrong}$.

Hybrid $\mathbf{H}_6$. In this hybrid,we modify the decryption phase. When for all $j \in [m]$ the mixer has sent $(\mathsf{VtDEC}, \mathsf{sid}, L, L_m)$ to $\mathcal{F}_{\mathsf{VtDec}}$, the hybrid simulates the answer of the ideal functionality sending the message $(\mathsf{sid}, \mathbf{b}, M'_o)$ where $\mathbf{b}$ is computed as defined by the ideal functionality $\mathcal{F}_{\mathsf{VtDec}}$ and M'_o is the empty list () if `Invalid` occurs; else, if all the messages in L correctly decrypt and the mixer proofs are valid, compute $M'_o \leftarrow (\mathtt{M}_{h^*-1,\zeta_o(j)})_{j\in[n]}$, where ζ_o is an uniformly random permutation. Notice that $\mathbf{H}_6$ does not use the map ψ^{-1}_{hide} at decryption phase.

We show that this hybrid and the previous one are equivalently distributed. First, by the change introduced in the previous hybrid, if the hybrid does not abort then $V_m = V_{h^*-1}$. Moreover, the two sets below are equivalently distributed:

$$\{(\mathtt{M}_{h^*-1,j}, \mathtt{H}_j) : j \in [n]\} \equiv \{(\mathtt{M}_{h^*-1,j}, \mathtt{H}_{\zeta_o(j)} : j \in [n])\}$$

because the messages $\mathtt{H}_1, \ldots, \mathtt{H}_n$ are uniformly distributed.

Hybrid $\mathbf{H}_7$. Similarly to what done in $\mathbf{H}_3$, in this hybrid we introduce the set Ψ_{in}, and we populate it with the pairs $(\mathtt{M}_i, \tilde{\mathtt{M}}_i)_{i\leq[n]}$, where the messages $\mathtt{M}_i$ are the inputs of the honest senders, and the messages $\tilde{\mathtt{M}}_i$ are distinct and sampled uniformly at random from the message space $\mathcal{M}$. When we simulate the ideal functionality $\mathcal{F}_{\mathsf{VtDec}}$, in case all the ciphertexts decrypts, we output the list $M_o :=(\mathtt{M}_{o,i})_i$, where $\mathtt{M}_{o,\zeta_o(i)} \leftarrow \psi^{-1}_{\text{in}}(\mathtt{M}_{h^*-1,i})$. We notice that if $V_{h^*-1} \cap \mathcal{M}_H \neq \emptyset$, the map ψ^{-1}_{in} would modify the returned value; however, since the messages $\tilde{\mathtt{M}}_i$ are not in the view of Z, there is a probability of at most $\frac{n^2}{|\mathcal{M}|}$ that this event happens and that Z distinguishes $\mathbf{H}_6$ from $\mathbf{H}_7$.

Hybrid $\mathbf{H}_8$. In this hybrid, we encrypt the simulated (honest) sender inputs $\tilde{\mathtt{M}}_j$ instead of the (honest) sender inputs $\mathtt{M}_j$ to populate the list L_0. The proof that this hybrid and the previous one are computationally indistinguishable follows by the lRCCA security of PKE and the zero-knowledge of $\mathsf{NIZK}_{\mathsf{sd}}$.

Lemma 3. *Hybrids $\boldsymbol{H}_7$ and $\boldsymbol{H}_8$ are computationally indistinguishable.*

Proof. First, we switch to hybrid $\mathbf{H}'_7$ and $\mathbf{H}'_8$ that are exactly the same but where the $\mathsf{crs}_{\mathsf{sd}}$ is sampled with $\mathsf{ABOInit}(j)$ for an arbitrary index j for a corrupted party. The hybrids can be shown indistinguishable based on the lRCCA-security of the scheme PKE, and the zero-knowledge of $\mathsf{NIZK}_{\mathsf{sd}}$. We can use a hybrid argument. Let $\mathbf{H}'_{7,j}$ be the hybrid game in which we encrypt the simulated sender inputs $\tilde{\mathtt{M}}_i$, for $i \leq j$, and we encrypt the honest sender inputs $\mathtt{M}_i$ for $i > j$. In particular, $\mathbf{H}'_7 \equiv \mathbf{H}'_{7,0}$ and $\mathbf{H}'_8 \equiv \mathbf{H}'_{7,n}$. We can prove that the hybrids $\mathbf{H}'_{7,j-1}$ is indistinguishable from $\mathbf{H}'_{7,j}, \forall j \in [n]$. In particular, when the j-th party is corrupt, the two hybrids are identically distributed. We focus on the more interesting case when the j-th sender party is honest.

Adversary $\mathcal{B}(\mathsf{pk})$ with oracle access to $\mathsf{ODec}(\cdot)$.

1. Start simulating the ideal functionality $\mathcal{F}_{\mathsf{VtDec}}$ sending the answer $(\mathsf{KEY}, \mathsf{sid}, \mathsf{pk})$ to the mixer parties, when instructed by the environment, thus embedding the public key from the challenger in the simulation.

2. When the honest sender party $\mathcal{P}_{S_i}$ is activated by Z on input $(\mathsf{INPUT}, \mathtt{M}_i)$, if $i < j$ sample a random message $\tilde{\mathtt{M}}_i$, encrypt $\tilde{\mathtt{M}}_i$, and add the pair $(\mathtt{M}_i, \tilde{\mathtt{M}}_i)$ to the set Ψ_{in}, and finally simulate the proof π^i_{sd}. Instead, if $i > j$, behave like in $\mathbf{H}_8$ encrypting the honest sender input $\mathtt{M}_i$. For $i = j$, sample a random message $\tilde{\mathtt{M}}_j$ and submit the pair $(\mathtt{M}_j, \tilde{\mathtt{M}}_j)$ to the lRCCA challenger, thus receiving back the challenge ciphertext $\mathtt{C}^*$; produce a simulated proof π^j_{sd} to be attached to $\mathtt{C}^*$ and continue the simulation.
3. When all the mixers have sent the message (VtDEC, L, L_m) to $\mathcal{F}_{\text{VtDec}}$, decrypt all of the ciphertexts in L by sending queries to the guarded decryption oracle, i,e. send the query $\mathtt{C}_{i',j}$, receiving back the message $\mathtt{M}_{i',j} \in \mathcal{M} \cup \{\diamond, \bot\}$ and if $\mathtt{M}_{i',j} = \diamond$ then set $\mathtt{M}_{i',j} := \psi^{-1}_{\text{in}}(\tilde{\mathtt{M}}_j)$. If a decryption error is returned for any of the queries, or any of the proofs attached to the ciphertexts are not valid (i.e. in case of $\mathtt{Invalid}$), abort and output a random bit b'. Else, simulate the leakage from $\mathcal{F}_{\mathsf{VtDec}}$ through the leakage received by the lRCCA security experiment; receive the value $f(\mathsf{sk})$ and send the message $(\mathsf{sid}, \mathbf{b}, (\mathtt{M}_{h^*-1,\zeta_o(j)})_{j\in[n]})$ as described by the hybrid.
4. Finally, when the simulation is complete, outputs the same as Z.

We notice that if the challenge bit b of the lRCCA game is equal to 0, the simulation of $\mathcal{B}$ offered to Z is identically distributed to the view in $\mathbf{H}'_{7,j-1}$, while if the challenge bit is 1, Z is given a view identically distributed to the one in $\mathbf{H}_{7,j}$. Also, whenever the event $\mathtt{Invalid}$ occurs, the reduction $\mathcal{B}$ outputs a random bit, so conditioning on $\mathtt{Invalid}$ the advantage in the lRCCA security game is equal to 0. We have that for an environment Z, $\mathbf{Adv}^{\mathtt{lRCCA}}_{\mathcal{B},\mathsf{PKE},f}(\lambda) = |\Pr[\mathbf{H}'_{7,j-1} = 1] - \Pr[\mathbf{H}_{7,j} = 1]|$.

We now introduce the latest two hybrids that ensure that none of the inputs of the honest senders is duplicated or discarded: we start by introducing a check on malicious senders, while in $\mathbf{H}_{10}$ we ensure that no malicious mixer can duplicate or discard the honest inputs.

Hybrid $\mathbf{H}_9$. Let $\mathcal{M}_H$ be the set of simulated messages $\{\tilde{\mathtt{M}}_i\}_{i\leq[n]}$ for the honest sender parties and let V_0 be the decryption of the list of ciphertexts received by the first mixer. If $\neg\mathtt{Invalid}$ and a message $\mathtt{M} \in \mathcal{M}_H$ appears more than once in the list V_0 then the simulation aborts.

Lemma 4. *Hybrids $\mathbf{H}_8$ and $\mathbf{H}_9$ are computationally indistinguishable.*

Proof. We rely on the lRCCA-security of PKE and the soundness of $\mathsf{NIZK}_{\mathsf{sd}}$, that indeed prevents the adversary from sending valid ciphertexts whose messages are correlated with the honest ones, to show that this abort happens only with negligible probability. Let $\mathbf{H}_{8,i}$ be the hybrid that aborts if $\neg\mathtt{Invalid}$ and a message $\mathtt{M} \in \mathcal{M}_H$ appears more than once in the list $(\mathtt{M}_{0,j})_{j\leq i}$. Clearly, $\mathbf{H}_{8,0} \equiv \mathbf{H}_8$ and $\mathbf{H}_{8,n} \equiv \mathbf{H}_9$. Next, let $\mathbf{H}'_{8,i}$ be the same as $\mathbf{H}_{8,i}$ but where the common reference string $\mathsf{crs}_{\mathsf{sd}}$ is sampled using $\mathsf{ABOInit}(i)$. We now show that $\mathbf{H}'_{8,i-1}$ is indistinguishable from $\mathbf{H}'_{8,i}$ for all $i \in [n]$. Notice that only difference between them is when one hybrid aborts while the other does not. Let $\mathtt{Bad}_i$ be the event that $(\neg\mathtt{Invalid} \wedge \mathtt{M}_{0,i} \in \mathcal{M}_H)$. It holds that:

$$|\Pr\big[\mathbf{H}'_{8,i-1}(\lambda) = 1\big] - \Pr\big[\mathbf{H}'_{8,i}(\lambda) = 1\big]| \leq n \cdot \Pr[\mathtt{Bad}_i].$$

We focus on the case where i is the index of a malicious sender, as the other case is obvious. We can show a reduction to the lRCCA security of PKE.

Adversary $\mathcal{B}(\mathsf{pk})$ with oracle access to $\mathsf{ODec}(\cdot)$.

1. Simulate the hybrid experiment $\mathbf{H}'_{8,i}$; in particular, when Z instructs a corrupted mixer to send the message $(\mathsf{KEY}, \mathsf{sid})$, simulate $\mathcal{F}_{\mathsf{VtDec}}$ sending back the answer $(\mathsf{KEY}, \mathsf{sid}, \mathsf{pk})$, embedding the public key from the challenger in the simulation. Also, sample $\mathsf{crs}_{\mathsf{sd}} \leftarrow\$ \mathsf{ABOInit}(i)$.
2. Sample $\tilde{\mathtt{M}}^{(0)}, \tilde{\mathtt{M}}^{(1)} \leftarrow\$ \mathcal{M}$ and send the pair of messages $(\tilde{\mathtt{M}}^{(0)}, \tilde{\mathtt{M}}^{(1)})$ to the lRCCA challenger, receiving back the challenge ciphertext $\mathtt{C}^*$.
3. Sample an index $h \in [n]$, such that the h-th sender is honest.
4. When the honest sender party $\mathcal{P}_{S_j}$ is activated on input $(\mathsf{INPUT}, \mathsf{sid}, \mathtt{M}_j)$, if $j \neq h$ sample a random message $\tilde{\mathtt{M}}_j$ (and populate the set Ψ_{in}), compute $\mathtt{C} \leftarrow\$ \mathsf{Enc}(\mathsf{pk}, \tilde{\mathtt{M}}_j)$, the honest proof π^j_{sd} (as in $\mathbf{H}_8$) and send to the other parties $(\mathsf{sid}, j, \mathtt{C}_j, \pi^j_{\mathsf{sd}})$. For $j = h$, instead, send $(\mathsf{sid}, h, \mathtt{C}^*, \tilde{\pi}^h_{\mathsf{sd}})$, where the proof $\tilde{\pi}^h_{\mathsf{sd}}$ is simulated. Wait for all the senders to broadcast their messages $(\mathsf{sid}, j, \mathtt{C}_j, \pi^j_{\mathsf{sd}})$ and continue the simulation.
5. When all the mixers have sent the message $(\mathsf{VtDECsid}, L, L_m)$, decrypt all of the ciphertexts in the list L by sending queries to the guarded decryption oracle, namely, send the query $\mathtt{C}_{k,j}$ for all $k \in [m]$ and all $j \in [n]$, receiving back as answer the plaintext messages $\mathtt{M}_{k,j} \in \mathcal{M} \cup \{\diamond, \perp\}$.
   If the event $\mathtt{Invalid}$ holds, then abort the simulation and output a random bit b'.
6. If $\mathtt{M}_{0,i} \neq \diamond$ then abort the simulation and output a random bit b'.
7. From now one we can assume that $\neg\mathtt{Invalid}$ and $\mathtt{M}_{0,i} \in \{\tilde{\mathtt{M}}^{(0)}, \tilde{\mathtt{M}}^{(1)}\}$; extract from the proof π^i_{sd} the plaintext message $\mathtt{M}$. Output b' s.t. $\mathtt{M} = \tilde{\mathtt{M}}^{(b')}$. If the extraction fails, output a random bit.

First, we notice that the simulation $\mathcal{B}$ provides to the environment Z is perfect: independently of the challenge bit b, the message $\tilde{\mathtt{M}}^{(b)}$ is distributed identically to $\tilde{\mathtt{M}}_h$, and the simulated proof is indistinguishable from the honest one, due to the ABO Zero-Knowledge property of $\mathsf{NIZK}_{\mathsf{sd}}$. Thus the probability that $\mathtt{Bad}_i$ happens in the reduction is the same as the probability the event happens in the hybrid experiments, conditioned on the fact that the guess of index h is correct (i.e. our strategy works with probability $\frac{1}{n}$).

With an analysis similar to what done in Lemma 2, we can easily prove that the event $\mathtt{Wrong}$, i.e. the fact that $\mathtt{M}_{0,i} = \tilde{\mathtt{M}}^{(1-b)}$, only happens with negligible probability $\frac{1}{|\mathcal{M}|}$. Conditioning on $\neg\mathtt{Wrong} \wedge \neg\mathtt{Abort}$, we have that $\mathcal{B}$ always outputs the correct bit $b' = b$.

Hybrid $\mathbf{H}_{10}$. Recall that $V_{h^*} \coloneqq (\mathtt{M}_{h^*,j})_{j\in[n]}$ is the list of decrypted ciphertexts output by the first honest mixer $\mathcal{P}_{M_{h^*}}$. In the hybrid $\mathbf{H}_{10}$ the simulation aborts if $\neg\mathtt{Invalid}$ and $\exists i \in [n]$ such that $\mathsf{Count}(\tilde{\mathtt{M}}_i, V_{h^*-1}) \neq 1$, i.e., some of the simulated honest inputs do not appear or appear more than once, encrypted, in the list received in input by the first honest mixer. With this check we ensure that none of the inputs of the honest senders has been discarded or duplicated by the (malicious) mixers.

Lemma 5. *Hybrids $\boldsymbol{H}_9$ and $\boldsymbol{H}_{10}$ are computationally indistinguishable.*

Proof. First, we switch to hybrid $\mathbf{H}'_9$ and $\mathbf{H}'_{10}$ that are exactly the same but where the $\mathsf{crs_{sd}}$ is sampled with $\mathsf{ABOInit}(j)$ for an arbitrary index j for a corrupted party.

We prove this using an hybrid argument. Let $\mathbf{H}_{9',i}$ the hybrid in which the simulation aborts if $\neg\mathtt{invalid}$ and $\exists j \leq i$ such that $\mathsf{Count}(\tilde{\mathtt{M}}_j, V_{h^*-1}) \neq 1$. Clearly we have that $\mathbf{H}'_9 \equiv \mathbf{H}'_{9,0}$ and $\mathbf{H}'_{10} \equiv \mathbf{H}'_{9,n}$. We now show that for all $i \in [n]$, the hybrid $\mathbf{H}'_{9,i-1}$ is indistinguishable from $\mathbf{H}'_{9,i}$. This is trivially true when the i-th sender is corrupted ($\mathbf{H}'_{9,i-1}$ and $\mathbf{H}'_{9,i}$ in this case are identically distributed).

Let $\mathtt{Bad}_i$ be the event that $(\neg\mathtt{Invalid} \wedge \mathsf{Count}(\tilde{\mathtt{M}}_i, V_{h^*-1}) = 0)$. It is easy to check that:

$$|\Pr\big[\mathbf{H}'_{9,i-1}(\lambda) = 1\big] - \Pr\big[\mathbf{H}'_{9,i}(\lambda) = 1\big]| \leq \Pr[\mathtt{Bad}_i].$$

In fact, the two hybrids are equivalent if the event $\mathtt{Bad}_i$ does not happen.

We define an adversary to the lRCCA security of PKE that makes use of the event above.

<u>Adversary $\mathcal{B}(\mathsf{pk})$ with oracle access to $\mathsf{ODec}(\cdot)$.</u>

1. Simulate the hybrid experiment $\mathbf{H}'_{9,i}$; in particular, when the environment instructs a corrupted mixer to send the message $(\mathsf{KEY}, \mathsf{sid})$, simulate the ideal functionality $\mathcal{F}_{\mathsf{VtDec}}$ sending back the answer $(\mathsf{KEY}, \mathsf{sid}, \mathsf{pk})$, embedding the public key from the challenger in the simulation.
2. Sample $\tilde{\mathtt{M}}^{(0)}, \tilde{\mathtt{M}}^{(1)} \leftarrow\$ \mathcal{M}$ and send the pair of messages $(\tilde{\mathtt{M}}^{(0)}, \tilde{\mathtt{M}}^{(1)})$ to the lRCCA challenger, receiving back the challenge ciphertext $\mathtt{C}^*$.
3. When the honest sender party $\mathcal{P}_{S_j}$ is activated on input $(\mathsf{INPUT}, \mathsf{sid}, \mathtt{M}_j)$, if $i \neq j$ sample a random message $\tilde{\mathtt{M}}_j$ (and populate the set Ψ_{in}), encrypt the message $\tilde{\mathtt{M}}_j$ as in $\mathbf{H}_{10}$ and continue the simulation by sending to the other parties $(\mathsf{sid}, j, \mathtt{C}_j, \pi^j_{\mathsf{sd}})$. For $j = i$, instead, send $(\mathsf{sid}, i, \mathtt{C}^*, \tilde{\pi}^j_{\mathsf{sd}})$, where the proof $\tilde{\pi}^j_{\mathsf{sd}}$ is simulated. Wait for all the senders to broadcast their messages $(\mathsf{sid}, j, \mathtt{C}_j, \pi^j_{\mathsf{sd}})$ and continue the simulation.
4. When all the mixers have sent their message (sid, L_j), decrypt all of the ciphertexts in those lists by sending queries to the guarded decryption oracle, namely, send the query $\mathtt{C}_{i',j}$ for all $i' \in [m]$ and all $j \in [n]$, receiving back as answer the plaintext messages $\mathtt{M}_{i',j} \in \mathcal{M} \cup \{\diamond, \bot\}$.
   If the event $\mathtt{Invalid}$ holds, then abort the simulation and output a random bit b'.

5. Let $C \leftarrow \mathsf{Count}(\diamond, V_{h^*-1})$, if $C = 1$ then abort the simulation and output a random bit b'.
6. From now one we can assume that $\neg\mathtt{Invalid}$ and $C \neq 1$;Compute

$$\mathtt{M} \leftarrow (C-1)^{-1} \cdot \left(\sum_{j \in [n], \mathtt{M}_{0,j} \neq \diamond} \mathtt{M}_{0,j} - \sum_{j \neq \zeta_{h^*-1}(i)} \mathtt{M}_{h^*-1,j} \right). \quad (3)$$

Output b' s.t. $\mathtt{M} = \tilde{\mathtt{M}}^{(b')}$.

First, we notice that the simulation $\mathcal{B}$ provides to the environment Z is perfect, indeed, independently of the challenge bit, the message $\tilde{\mathtt{M}}^{(b)}$ is distributed identically to $\tilde{\mathtt{M}}_j$. Thus the probability that $\mathtt{Bad}_i$ happens in the reduction is the same as the probability the event happens in the hybrid experiments.

Let $\mathtt{Abort}$ be the event that $\mathcal{B}$ aborts and outputs a random bit. Notice that:

$$\mathtt{Abort} \equiv \mathtt{Invalid} \vee (C = 1).$$

Let $\mathtt{Wrong}$ be the event that $\exists j : \mathsf{Dec}(\mathsf{sk}, \mathtt{C}_{h^*-1,j}) = \tilde{\mathtt{M}}^{(1-b)}$, notice that the message $\tilde{\mathtt{M}}^{(1-b)}$ is independent of the view of the environment Z, thus the probability of $\mathtt{Wrong}$ is at most $n/|\mathcal{M}|$. Moreover, we have $\mathtt{Bad}_i \equiv \neg\mathtt{Abort} \wedge \neg\mathtt{Wrong}$ because, by definition of $\neg\mathtt{Wrong}$, all the ciphertexts that decrypt to $\diamond$ in L_{h^*-1} are indeed an encryption of $\tilde{\mathtt{M}}^{(b)}$, thus assuming the event holds then $C \neq 1$ iff $\mathsf{Count}(\tilde{\mathtt{M}}^{(b)}, V_{h^*-1}) \neq 1$. The probability of guessing the challenge bit when $\mathcal{B}$ aborts is $\frac{1}{2}$, thus we have:

$$\Pr[b = b'] \geq \tfrac{1}{2} \Pr[\neg\mathtt{Bad}_i] + \Pr[b = b' | \mathtt{Bad}_i] \Pr[\mathtt{Bad}_i] - \tfrac{n}{|\mathcal{M}|} \quad (4)$$

We now compute the probability that $b = b'$ conditioned on the event $\mathtt{Bad}_i$. First, we notice that $\neg\mathtt{Invalid}$ implies that the ciphertexts in the lists $L_0, \ldots, L_{h^*-1}$ decrypt correctly and that the proofs π^j_{mx} for $j < h^*$ verify. Thus by applying the sumcheck-admissibility w.r.t. PKE of the relation $\mathcal{R}_{\mathsf{mx}}$ and by the ABO perfect soundness of $\mathsf{NIZK}_{\mathsf{mx}}$ we have:

$$\sum_{j \in [n]} \mathsf{Dec}(\mathsf{sk}, \mathtt{C}_{0,j}) - \sum_{j \in [n]} \mathsf{Dec}(\mathsf{sk}, \mathtt{C}_{h^*-1,j}) = 0.$$

If we condition on $\neg\mathtt{Wrong}$ then:

$$\left(\tilde{\mathtt{M}}^{(b)} + \sum_{j \neq \zeta_0(j^*)} \mathtt{M}_{0,j} \right) - \left(C \cdot \tilde{\mathtt{M}}^{(b)} + \sum_{j \in [n], \mathtt{M}_{h^*-1,j} \neq \diamond} \mathtt{M}_{h^*-1,j} \right) = 0.$$

By solving the above equation for $\tilde{\mathtt{M}}^{(b)}$, we obtain that $\mathtt{M} = \tilde{\mathtt{M}}^{(b)}$. We recall that the index $\tau^* \in [n]$ is (only) computationally hidden: $\mathcal{B}$ is able to extract from the proof $\pi^{i'}_{\mathsf{sd}}$ with probability $\frac{1}{n} - \mathsf{negl}(\lambda)$. Therefore, $\mathcal{B}$ guesses the challenge bit with probability 1 when conditioning on $\neg\mathtt{Abort} \wedge \neg\mathtt{Wrong} \wedge \tau^* = i'$.

Simulator S.

Initialization. Simulate the ideal functionality $\mathcal{F}_{\mathsf{crs}}$ by sampling $\mathsf{crs}_{\mathsf{mx}}$ in ABO Perfect Sound mode on the tag h^*, while $\mathsf{crs}_{\mathsf{sd}}$ is honestly generated with $\mathsf{Init}(1^\lambda)$. Also, simulate $\mathcal{F}_{\mathsf{VtDec}}$ by a sampling key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{KGen}(\mathsf{prm})$. Populate the set $\mathcal{M}_H$ of the simulated honest inputs, by sampling uniformly random (and distinct) messages from the message space $\mathcal{M}$.

Honest Senders. On activation of the honest sender $\mathcal{P}_{S_i}$, where $i \in [n]$, simulate by executing the code of the honest sender on input the simulated message $\tilde{\mathtt{M}}_j$ chosen uniformly at random, without re-introduction, from $\mathcal{M}_H$.

Extraction of the Inputs. Let L_{h^*-1} be the list produced by the malicious mixer $\mathcal{P}_{M_{h^*-1}}$. For any $j \in [n]$, decrypt $\mathtt{M}_j \leftarrow\$ \mathsf{Dec}(\mathsf{sk}, \mathtt{C}_{h^*-1,j})$ and if a decryption error occurs, or some of the mixer proofs π^j_{mx} is not valid, i.e. the event `Invalid` occurs, abort the simulation. If $\mathtt{M}_j \notin \mathcal{M}_H$ then submit it as input to the ideal functionality $\mathcal{F}_{\mathrm{Mix}}$.

First Honest Mixer. Simulate by computing L_{h^*} as a list of encryption of random (distinct) messages $\mathtt{H}_1, \ldots, \mathtt{H}_n$, simulating the proof of mixing $\pi^{h^*}_{\mathsf{mx}}$.

Verification Phase. Receive from the ideal mixer functionality $\mathcal{F}_{\mathrm{Mix}}$ the sorted output $(\mathtt{M}_i)_{i\in[n]}$. Sample a random permutation ζ_o and populate the list of outputs $M_o \coloneqq (\mathtt{M}_{o,i})_{i\in[n]}$ with $\mathtt{M}_{o,\zeta_o(i)} \leftarrow \mathtt{M}_i$.

We notice that there are some differences between $\mathbf{H}_{10}$ and the interaction of S with the ideal functionality $\mathcal{F}_{\mathrm{Mix}}$. In particular, the hybrid defines the function ψ_{in} by setting a mapping between the inputs of the honest senders and the simulated ones, and, during the decryption phase, and uses ψ_{in}^{-1} to revert this change. S cannot explicitly set this mapping, because the inputs of the honest senders are sent directly to the functionality and are unknown to S. However, the simulator is implicitly defining the function ψ_{in} (and ψ_{in}^{-1}) since during the simulation chooses a simulated input $\tilde{\mathtt{M}}_i$ for each honest sender and at decryption phase outputs the messages coming from the sorted list (given in output by the ideal functionality) which contains the inputs of the honest senders.

5 A concrete Mix-Net protocol from RCCA-PKE

As already mentioned, to instantiate the blue-print protocol defined in Fig. 3 we need two main components: (1) a Rand lRCCA PKE scheme PKE and (2) a verify-then-decrypt protocol for such PKE.

5.1 Split PKE

We start by introducing the notion of Split Public-Key Encryption scheme. Informally, a Split PKE scheme is a special form of PKE scheme that extends and builds upon another PKE scheme. For example, CCA-secure PKE schemes alá Cramer-Shoup [CS98] can be seen as an extension of CPA-secure PKE schemes. We give the formal definition in the following.

Definition 6 (Split PKE). *A split PKE scheme* PKE *is a tuple of seven randomized algorithms:*

$\mathsf{Setup}(1^\lambda)$: *upon input the security parameter* 1^λ *produces public parameters* prm*, which include the description of the message* ($\mathcal{M}$) *and two ciphertext spaces* ($\mathcal{C}_1, \mathcal{C}_2$)*.*

$\mathsf{KGen}_A(\mathsf{prm})$: *upon input the parameters* prm*, outputs a key pair* $(\mathsf{pk}_A, \mathsf{sk}_A)$*.*

$\mathsf{KGen}_B(\mathsf{prm}, \mathsf{pk}_A)$: *upon inputs the parameters* prm *and a previously generated public key* pk_A*, outputs a key pair* $(\mathsf{pk}_B, \mathsf{sk}_B)$*.*

$\mathsf{Enc}_A(\mathsf{pk}_A, \mathtt{M}; r)$: *upon inputs a public key* pk_A*, a message* $\mathtt{M} \in \mathcal{M}$*, and randomness* r*, outputs a ciphertext* $\mathtt{C}_A \in \mathcal{C}_A$*.*

$\mathsf{Enc}_B(\mathsf{pk}_A, \mathsf{pk}_B, \mathtt{C}; r)$: *upon inputs a pair of public keys* $(\mathsf{pk}_A, \mathsf{pk}_B)$*, a ciphertext* $\mathtt{C} \in \mathcal{C}_A$*, and some randomness* r*, outputs a ciphertext* $\mathtt{C}_B \in \mathcal{C}_B$*.*

$\mathsf{Dec}_A(\mathsf{pk}_A, \mathsf{sk}_A, \mathtt{C})$: *upon inputs a secret key* sk_A *and a ciphertext* $\mathtt{C} \in \mathcal{C}_A$*, outputs a message* $\mathtt{M} \in \mathcal{M}$ *or an error symbol* $\perp$*.*

$\mathsf{Dec}_B(\mathsf{pk}_A, \mathsf{pk}_B, \mathsf{sk}_A, \mathsf{sk}_B, \mathtt{C})$: *upon inputs secret keys* $\mathsf{sk}_A, \mathsf{sk}_B$ *and a ciphertext* $\mathtt{C} \in \mathcal{C}_B$*, outputs a message* $\mathtt{M} \in \mathcal{M}$ *or an error symbol* $\perp$*.*

Moreover, we say that a split PKE scheme PKE splits on *a PKE scheme* $\mathsf{PKE}_A :=$ $(\mathsf{KGen}_A, \mathsf{Enc}_A, \mathsf{Dec}_A)$ *defined over message space* $\mathcal{M}$ *and ciphertext space* $\mathcal{C}_A$ *and we say that a split PKE scheme* PKE forms *a PKE* $\mathsf{PKE} := (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *defined over message space* $\mathcal{M}$ *and ciphertext space* $\mathcal{C}_B$ *where* $\mathsf{KGen}(\mathsf{prm})$ *is the algorithm that first runs* $\mathsf{pk}_A, \mathsf{sk}_A \leftarrow\$ \mathsf{KGen}_A(\mathsf{prm})$*, then runs* $\mathsf{pk}_B, \mathsf{sk}_B \leftarrow\$$ $\mathsf{KGen}_B(\mathsf{prm}, \mathsf{pk}_A)$ *and sets* $\mathsf{pk} := (\mathsf{pk}_A, \mathsf{pk}_B)$*,* $\mathsf{sk} := (\mathsf{sk}_A, \mathsf{sk}_B)$*, where* $\mathsf{Enc}(\mathsf{pk}, \mathtt{M})$ *is the algorithm that outputs* $\mathsf{Enc}_B(\mathsf{pk}_A, \mathsf{pk}_B, \mathsf{Enc}_A(\mathsf{pk}_A, \mathtt{M}; r); r)$ *and* $\mathsf{Dec} := \mathsf{Dec}_B$*.*

The correctness property is straightforward: a split PKE is correct if it forms a PKE that is correct in the standard sense. Our definition is general enough to capture a large class of schemes. We first note that any PKE scheme is trivially split: it suffices that Enc_B on input $\mathtt{C}$ outputs $\mathtt{C}$, and Dec_B runs Dec_A. A more natural (and less trivial) example is the above-cited Cramer-Shoup.

In this paper, we will focus on PKE schemes that are Re-Randomizable and Verifiable. Since, as we noted above, any PKE can be parsed as a Split PKE, Re-Randomizability is captured by an additional algorithm $\mathsf{Rand}(\mathsf{pk}, \mathtt{C}; r)$ that takes as input a ciphertext $\mathtt{C}$ and outputs a new ciphertext $\hat{\mathtt{C}}$ (see Section 2.1).

As for the verifiability property, instead, there are three possible levels: (i) both the secret keys are required to verify a ciphertext, or (ii) only sk_A is needed, or (iii) no secret key is required at all. We refer to the third one as the *public* setting, while the other two are different flavors of a private/designated-verifier setting. We give the definition of (ii) in what follows.

Definition 7 (verifiable split PKE). *A verifiable split PKE is a split PKE, as defined above, with an additional algorithm* $\mathsf{Vf}(\mathsf{pk}, \mathsf{sk}_B, \mathtt{C})$ *that takes as input the public key* pk*, the secret key* sk_B *and a ciphertext* $\mathtt{C} \in \mathcal{C}_B$ *and outputs* 1 *whenever* $\mathsf{Dec}_B(\mathsf{pk}, \mathsf{sk}, \mathtt{C}) \neq \perp$*, otherwise outputs* 0 *for invalid ciphertexts.*

Functionality $\mathcal{F}_{\mathsf{Dec}}^{\mathsf{PKE}}$

The ideal functionality has as parameters a public-key encryption scheme $\mathsf{PKE} := (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ and (implicit) public parameters prm. The functionality interacts with m parties $\mathcal{P}_i$ and with an adversary S.

Public Key. Upon activation on message $(\mathsf{KEY}, \mathsf{sid})$ from a party $\mathcal{P}_i$, $i \in [m]$, if $(\mathsf{sid}, \mathsf{pk}, \mathsf{sk})$ is not in the database sample $(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{KGen}(\mathsf{prm})$ and store the tuple $(\mathsf{sid}, \mathsf{pk}, \mathsf{sk})$ in the database and send $(\mathsf{KEY}, \mathsf{sid}, \mathsf{pk})$ to $\mathcal{P}_i$.

Decryption. Upon activation on $(\mathsf{DECRYPT}, \mathsf{sid}, C)$ from party $\mathcal{P}_i, i \in [m]$:
- If the tuple $(\mathsf{sid}, \mathsf{pk}, \mathsf{sk})$ does not exist in the database, ignore the message.
- Check that a tuple $(\mathsf{sid}, C, M_o, \mathcal{I})$, where $\mathcal{I} \subseteq [m]$, exists in the database; if so, update $\mathcal{I}$ including the index i. Else, parse C as $(\mathsf{C}_i)_i$ and compute the list $M_o := (\mathsf{Dec}(\mathsf{sk}, \mathsf{C}_i))_{i \in [|C|]}$, and create the new entry $(\mathsf{sid}, C, M_o, \{i\})$ in the database.
- If $|\mathcal{I}|$ equals m, then send a public delayed output $(\mathsf{DECRYPT}, \mathsf{sid}, M_o)$ to the parties $\mathcal{P}_i$ for $i \in [m]$.

Fig. 6. UC ideal functionality for (n-out-n Threshold) Key-Generation and Decryption of PKE

5.2 A protocol for Verify-then-Decrypt for verifiable split PKE

We realize the Verify-then-Decrypt ideal functionality (see Section 3.2) needed to instantiate our Mix-Net protocol. Let PKE be a verifiable split PKE. We define in Fig. 8 the protocol Π_{VtDec} that realizes $\mathcal{F}_{\mathsf{VtDec}}$ in the $\mathcal{F}_{\mathsf{Com}}$-hybrid model. Before doing that, we need to assume an extra property for our verifiable split PKE, so we introduce the notion of linear key-homomorphism for a PKE.

Definition 8 (Linearly Key-Homomorphic PKE). *We say that a PKE* $\mathsf{PKE} := (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *is linearly key-homomorphic if there exist PPT algorithms* $\mathsf{GenPK}, \mathsf{CheckPK}$ *and an integer s such that:*

- *The algorithm* $\mathsf{KGen}(\mathsf{prm})$*, where* prm *contains the description of a group of order q, first executes* $\mathsf{sk} \leftarrow\$ \mathbb{Z}_q^s$*, and then produces the public key* $\mathsf{pk} \leftarrow\$ \mathsf{GenPK}(\mathsf{sk})$.
- *The algorithm* GenPK *is linearly homomorphic in the sense that for any* $\mathsf{sk}_1, \mathsf{sk}_2 \in \mathbb{Z}_q^s$ *and* $\alpha \in \mathbb{Z}_q^s$ *we have* $\mathsf{GenPK}(\alpha \cdot \mathsf{sk}_1 + \mathsf{sk}_2) = \alpha \cdot \mathsf{GenPK}(\mathsf{sk}_1) + \mathsf{GenPK}(\mathsf{sk}_2)$.
- *The algorithm* $\mathsf{CheckPK}$ *on input the public key* pk *outputs a bit b to indicate if the public key belongs on the subgroup of $\mathcal{PK}$ spanned by* GenPK*. Namely, for any* pk *we have* $\mathsf{CheckPK}(\mathsf{pk}) = 1$ *iff* $\mathsf{pk} \in Im(\mathsf{GenPK}(\mathsf{prm}, \cdot))$.

Moreover, a split PKE PKE *is linearly key-homomorphic it forms a linearly key-homomorphic PKE and it splits to a key-homomorphic PKE.*

It is not hard to verify that the key generation of a linearly key-homomorphic split PKE can be seen as sampling two secret vectors $\mathsf{sk}_A \in \mathbb{Z}_q^s$ and $\mathsf{sk}_B \in \mathbb{Z}_q^{s'}$

for $s, s' \in \mathbb{N}$ and then applying two distinct homomorphisms $\mathsf{GenPK}_A, \mathsf{GenPK}_B$ to derive the public key.

Building Blocks. Let PKE be a split PKE that splits over PKE_A, consider the following building blocks:

1. An ideal functionality $\mathcal{F}_{\mathsf{Dec}}^{\mathsf{PKE}_A}$ for threshold decryption, as defined in Fig. 6, of PKE_A.
2. A single-sender multiple-receiver commitment ideal functionality $\mathcal{F}_{\mathsf{Com}}$ [CF01] for strings, as defined in Fig. 7.

We describe the protocol in Fig. 8. At a high level, the protocol works as follows. Each party $\mathcal{P}_i$ interacts with the ideal functionality $\mathcal{F}_{\mathsf{Dec}}$ to get the public key pk_A and, after that, samples the pair of keys $(\mathsf{pk}_B^i, \mathsf{sk}_B^i)$. The secret key is committed through the ideal functionality $\mathcal{F}_{\mathsf{Com}}$. After this step, the parties compute the final key pk_B as the sum of all their input public key shares. To verify the ciphertexts C_V, the parties reveal their secret key shares sk_B^i, verify that all the keys are consistent, and locally verify the ciphertexts. Finally, to decrypt the ciphertexts C_D, the parties invoke $\mathcal{F}_{\mathsf{Dec}}$ after checking that $C_D \subseteq C_V$.

Functionality $\mathcal{F}_{\mathsf{Com}}$

The functionality interacts with n parties $\mathcal{P}_i$ and an adversary S.

Commitment. Upon activation on message $(\mathsf{COMMIT}, \text{sid}, \mathcal{P}_i, s)$ from a party $\mathcal{P}_i$, where $s \in \{0,1\}^*$, record the tuple $(\text{sid}, \mathcal{P}_i, s)$ and send the public delayed output $(\mathsf{RECEIPT}, \text{sid}, \mathcal{P}_i)$ to all the parties $\mathcal{P}_j$, $j \in [n], j \neq i$.

Opening. Upon activation on message $(\mathsf{OPEN}, \text{sid}, \mathcal{P}_i)$ from a party $\mathcal{P}_i$, $i \in [n]$, proceed as follows: if the tuple $(\text{sid}, \mathcal{P}_i, s)$ was previously recoded, then send the public delayed output $(\mathsf{OPEN}, \text{sid}, \mathcal{P}_i, s)$ to all other parties $\mathcal{P}_j, j \in [n], j \neq i$. Otherwise halt.

Fig. 7. UC ideal functionality for (Single) Commitment.

Theorem 2. *Let* PKE *be a verifiable split PKE that is linearly key-homomorphic, let* f *be the leakage function that on input* $\mathsf{sk} := (\mathsf{sk}_A, \mathsf{sk}_B)$ *outputs* sk_B. *The protocol* $\Pi_{\mathsf{VtDec}}^{\mathsf{PKE}}$ *described in Fig. 8 UC-realizes the functionality* $\mathcal{F}_{\mathsf{VtDec}}^{\mathsf{PKE},f}$ *described in Fig. 2 with setup assumptions* $\mathcal{F}_{\mathsf{Dec}}^{\mathsf{PKE}_A}$ *and* $\mathcal{F}_{\mathsf{Com}}$.

Proof. We now prove that there exists a simulator S such that no PPT environment Z can distinguish an interaction with the real protocol from an interaction with S and the ideal functionality $\mathcal{F}_{\mathsf{VtDec}}$.

Simulator S.

Protocol $\Pi_{\mathsf{VtDec}}^{\mathsf{PKE}}$

The party $\mathcal{P}_i$ executes the following commands:

Public Key. Upon activation on message:
- $(\mathsf{KEY}, \mathsf{sid})$ from the environment, forward the message to $\mathcal{F}_{\mathsf{Dec}}^{\mathsf{PKE}_A}$.
- $(\mathsf{KEY}, \mathsf{sid}, \mathsf{pk}_A)$ from $\mathcal{F}_{\mathsf{Dec}}^{\mathsf{PKE}_A}$ proceed as below:
  1. Sample $\mathsf{sk}_B^i \leftarrow\$ \mathbb{Z}_q^s$ compute $\mathsf{pk}_B^i \leftarrow \mathsf{GenPK}(\mathsf{sk}_B^i)$.
  2. Commit the secret key sk_B^i through the ideal functionality $\mathcal{F}_{\mathsf{Com}}$, i.e. send the message $(\mathsf{COMMIT}, \mathsf{sid}, \mathsf{sk}_B^i)$ to the functionality $\mathcal{F}_{\mathsf{Com}}$.
- $(\mathsf{RECEIPT}, \mathsf{sid}, \mathcal{P}_j)$ from all $j \in [m]$ broadcast $(\mathsf{KEY}, \mathsf{sid}, i, \mathsf{pk}_B^i)$.

When the parties have sent their public key shares, compute $\mathsf{pk}_B \coloneqq \sum_i \mathsf{pk}_B^i$ and abort if $\exists i : \mathsf{CheckPK}(\mathsf{pk}_A, \mathsf{pk}_B^i) = 0$ else output $(\mathsf{KEY}, \mathsf{sid}, \mathsf{pk})$.

Verify then Decrypt. Upon activation on message:
- $(\mathsf{VTDEC}, \mathsf{sid}, C_V, C_D)$ send $(\mathsf{OPEN}, \mathsf{sid}, \mathcal{P}_i)$ to $\mathcal{F}_{\mathsf{Com}}$ and broadcast $(\mathsf{VTDEC}, \mathsf{sid}, C_V, C_D)$ to the other parties.
- $(\mathsf{OPEN}, \mathsf{sid}, \mathcal{P}_j, \mathsf{sk}_B^j)$ for all $i \in [m]$ compute $\mathsf{sk}_B \coloneqq \sum_i \mathsf{sk}_B^j$ and assert that $\mathsf{GenPK}_B(\mathsf{sk}_B) \stackrel{?}{=} \mathsf{pk}_B$ and that all parties broadcast the same lists C_V and C_D. Parse C_V as $(\mathsf{c}_V^i)_{i \in |\mathcal{C}_V|}$, compute $\forall j : b_j \leftarrow \mathsf{Vf}(\mathsf{pk}, \mathsf{sk}_B, \mathsf{c}_V^j)$.
  If $C_D \not\subseteq C_V$ or $\exists i : b_i = 0$ return $(\mathsf{DECRYPT}, \mathsf{sid}, \mathbf{b}, ())$ else send $(\mathsf{DECRYPT}, \mathsf{sid}, C_D)$ to $\mathcal{F}_{\mathsf{Dec}}^{\mathsf{PKE}_A}$ and upon receipt of $(\mathsf{DECRYPT}, \mathsf{sid}, M_o)$, output $(\mathsf{DECRYPT}, \mathsf{sid}, \mathbf{b}, M_o)$

Fig. 8. Our protocol Π_{VtDec}.

Public Key. The simulator S receives in input from Z the set of corrupted parties, and receives from $\mathcal{F}_{\mathsf{VtDec}}$ the public key pk that is parsed as the tuple $(\mathsf{pk}_A, \mathsf{pk}_B)$. S gets to see the secret key shares of the corrupted parties when they send the message $(\mathsf{COMMIT}, \mathsf{sid}, \mathsf{sk}_B^i)$. Let h^* be the index of an honest party. S samples at random the secret keys sk_B^i for all honest parties $\mathcal{P}_i$, with $i \neq h^*$, from which can honestly compute the corresponding public keys through GenPK. As for the h^*-th party, S checks if $\forall j \neq h^* : \mathsf{CheckPK}(\mathsf{pk}_A, \mathsf{pk}_B^j) = 1$. If so it computes directly the public key $\mathsf{pk}_B^{h^*} \coloneqq \mathsf{pk}_B - \sum_{i \neq h^*} \mathsf{pk}_B^i$, else it samples $\mathsf{sk}_B^{h^*}$ and computes the corresponding public key.

Verification. When all the parties have sent the message $(\mathsf{OPEN}, \mathsf{sid}, \mathcal{P}_i)$ to the commitment functionality $\mathcal{F}_{\mathsf{Com}}$, the simulator receives the leakage $(\mathsf{sid}, \mathsf{sk}_B)$ from $\mathcal{F}_{\mathsf{VtDec}}^{\mathsf{PKE},f}$, it computes the secret key for party $\mathcal{P}_{h^*}$, i.e. it computes $\mathsf{sk}_B^{h^*} \coloneqq \mathsf{sk}_B - \sum_{i \neq h^*} \mathsf{sk}_B^i$. From this point on, the simulation becomes trivial since the simulator follows the protocol, and can easily verify and decrypt all the ciphertexts by interacting with the ideal functionality $\mathcal{F}_{\mathsf{VtDec}}$.

We observe that the inputs simulated for the honest parties $\mathcal{P}_i$, for $i \neq h^*$, are perfectly simulated since S chooses uniformly at random the matrices and the vectors for the secret keys sk_B^i. The public key for the h^*-th party is chosen

$\mathsf{KGen}_A(\mathsf{prm})$

$\mathbf{D} \leftarrow\!\!\$\ \mathcal{D}_k;\quad \mathbf{a} \leftarrow\!\!\$\ \mathbb{Z}_q^{k+1}$

$\mathbf{D}^* \leftarrow (\mathbf{D}^\top, (\mathbf{a}^\top \mathbf{D})^\top)^\top$

$\mathsf{sk}_A \leftarrow \mathbf{a};\quad \mathsf{pk}_A \leftarrow ([\mathbf{D}]_1, [\mathbf{a}^\top \mathbf{D}]_1)$

return $(\mathsf{pk}_A, \mathsf{sk}_A)$

$\mathsf{KGen}_B(\mathsf{prm}, \mathsf{pk}_A)$

$\mathbf{E} \leftarrow\!\!\$\ \mathcal{D}_k;\quad \mathbf{f}, \mathbf{g} \leftarrow\!\!\$\ \mathbb{Z}_q^{k+1}$

$\mathbf{F} \leftarrow\!\!\$\ \mathbb{Z}_q^{k+1\times k+1}, \mathbf{G} \leftarrow\!\!\$\ \mathbb{Z}_q^{k+1\times k+2}$

$\mathsf{sk}_B \leftarrow (\mathbf{f}, \mathbf{g}, \mathbf{F}, \mathbf{G})$

$\mathsf{pk}_B \leftarrow ([\mathbf{E}]_2, [\mathbf{f}^\top \mathbf{D}]_T, [\mathbf{F}^\top \mathbf{D}]_1,$
$\quad [\mathbf{g}^\top \mathbf{E}]_T, [\mathbf{G}^\top \mathbf{E}]_2, [\mathbf{G}\mathbf{D}^*]_1, [\mathbf{F}\mathbf{E}]_2)$

return $(\mathsf{pk}_B, \mathsf{sk}_B)$

$\mathsf{Dec}_A(\mathsf{pk}_A, \mathsf{sk}_A, \mathtt{C} = [\mathbf{x}]_1)$

return $[p]_1 - [\mathbf{a}^\top \mathbf{u}]_1$

$\mathsf{Enc}_A(\mathsf{pk}_A, [\mathtt{M}]_1; \mathbf{r})$

$[\mathbf{u}]_1 \leftarrow [\mathbf{D}]_1 \cdot \mathbf{r};\quad [p]_1 \leftarrow [\mathbf{a}^\top \mathbf{D}]_1 \cdot \mathbf{r} + [\mathtt{M}]_1$

return $([\mathbf{u}^\top]_1, [p]_1)^\top$

$\mathsf{Enc}_B(\mathsf{pk}, \mathtt{C} = [\mathbf{x}]_1; (\mathbf{r}, \mathbf{s}))$

$[\mathbf{v}]_2 \leftarrow [\mathbf{E}]_2 \cdot \mathbf{s}$

$[\pi_1]_T \leftarrow [\mathbf{f}^\top \mathbf{D}]_T \cdot \mathbf{r} + e([\mathbf{F}^\top \mathbf{D}]_1 \cdot \mathbf{r}, [\mathbf{v}]_2)$

$[\pi_2]_T \leftarrow [\mathbf{g}^\top \mathbf{E}]_T \cdot \mathbf{s} + e([\mathbf{x}]_1, [\mathbf{G}^\top \mathbf{E}]_2 \cdot \mathbf{s})$

$[\pi]_T \leftarrow [\pi_1]_T + [\pi_2]_T$

return $([\mathbf{x}]_1, [\mathbf{v}]_2, [\pi]_T)$

$\mathsf{Dec}_B(\mathsf{pk}, \mathsf{sk}, \mathtt{C} = ([\mathbf{x}]_1, [\mathbf{v}]_2, [\pi]_T))$

$[\pi_1]_T \leftarrow [(\mathbf{f} + \mathbf{F}\mathbf{v})^\top \mathbf{u}]_T$

$[\pi_2]_T \leftarrow [(\mathbf{g} + \mathbf{G}\mathbf{x})^\top \mathbf{v}]_T$

if $[\pi]_T \neq [\pi_1]_T + [\pi_2]_T$ **return** $\perp$

else return $\mathsf{Dec}_A(\mathsf{sk}_A, [\mathbf{x}]_1)$

$\mathsf{Rand}(\mathsf{pk}, \mathtt{C} = ([\mathbf{x}]_1, [\mathbf{v}]_2, [\pi]_T))$

parse $[\mathbf{x}]_1$ as $([\mathbf{u}^\top]_1, [p]_1)^\top$, $\quad \hat{\mathbf{r}}, \hat{\mathbf{s}} \leftarrow\!\!\$\ \mathbb{Z}_q^k$

$[\hat{\mathbf{x}}]_1 \leftarrow [\mathbf{x}]_1 + [\mathbf{D}^*]_1 \cdot \hat{\mathbf{r}},\quad [\hat{\mathbf{v}}]_2 \leftarrow [\mathbf{v}]_2 + [\mathbf{E}]_2 \cdot \hat{\mathbf{s}}$

$[\hat{\pi}_1]_T \leftarrow [\mathbf{f}^\top \mathbf{D}]_T \cdot \hat{\mathbf{r}} + e([\mathbf{F}^\top \mathbf{D}]_1 \cdot \hat{\mathbf{r}}, [\hat{\mathbf{v}}]_2) + e([\mathbf{u}]_1, [\mathbf{F}\mathbf{E}]_2 \cdot \hat{\mathbf{s}})$

$[\hat{\pi}_2]_T \leftarrow [\mathbf{g}^\top \mathbf{E}]_T \cdot \hat{\mathbf{s}} + e([\hat{\mathbf{x}}]_1, [\mathbf{G}^\top \mathbf{E}]_2 \cdot \hat{\mathbf{s}}) + e([\mathbf{G}\mathbf{D}^*]_1 \cdot \hat{\mathbf{r}}, [\mathbf{v}]_2)$

$[\hat{\pi}] \leftarrow [\pi]_T + [\hat{\pi}_1]_T + [\hat{\pi}_2]_T$

return $([\hat{\mathbf{x}}]_1, [\hat{\mathbf{v}}]_2, [\hat{\pi}]_T)$

Fig. 9. The Split RCCA-secure Scheme. prm include the description of a bilinear group.

dependently of the message of the corrupted parties. In particular, if one of the corrupted parties sends an invalid public key the h^*-th mixer follows the specification of the protocol, thus the simulation is perfect; if all the public keys are valid, the public key of h^*-th party is chosen as a function of the previously chosen keys and the public key given in input to the simulator. This is distributed identically to a real execution of the protocol: the only difference is that S computes the random public key, while in the real execution the party $\mathcal{P}_{h^*}$ would choose at random their secret key and then project it to compute the corresponding public key, but this difference is only syntactical. In the next steps, the simulation is perfect since it proceeds exactly as in the real protocol.

5.3 Our concrete verifiable split PKE

In this section, we show that the Rand-PKE in [FFHR19] has all the properties needed to instantiate our protocol Π_{Mix}. In particular, in Fig. 9 we parse their PKE as a split PKE, and we prove that the scheme is lRCCA w.r.t. the leakage function f such that $f(\mathsf{sk}) \coloneqq \mathsf{sk}_B$, and that the scheme is linearly key-homomorphic. Moreover, we show a checksum-admissible relation $\mathcal{R}_{\mathsf{mx}}$ w.r.t. the PKE, we show how to instantiate the ABO perfect sound tag-based NIZK $\mathsf{NIZK}_{\mathsf{mx}}$ using the Kiltz-Wee quasi-adaptive (QA) NIZK [KW15], and we instantiate the NIZK $\mathsf{NIZK}_{\mathsf{sd}}$ using GS-Proofs. The schemes in [FFHR19] are proven secure under a decisional assumption that we briefly introduce here. Let ℓ, k be two positive integers. We call $\mathcal{D}_{\ell,k}$ a matrix distribution if it outputs (in probabilistic polynomial time, with overwhelming probability) matrices in $\mathbb{Z}_q^{\ell \times k}$.

Definition 9 (Matrix Decisional Diffie-Hellman Assumption in $\mathbb{G}_\gamma$, [EHK+13]). *The $\mathcal{D}_{\ell,k}$-MDDH assumption holds if for all non-uniform PPT adversaries $\mathcal{A}$,*

$$|\Pr[\mathcal{A}(\mathcal{G}, [\mathbf{A}]_\gamma, [\mathbf{Aw}]_\gamma) = 1] - \Pr[\mathcal{A}(\mathcal{G}, [\mathbf{A}]_\gamma, [\mathbf{z}]_\gamma) = 1]| \in \mathsf{negl}(\lambda),$$

where the probability is taken over $\mathcal{G} \coloneqq (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathcal{P}_1, \mathcal{P}_2) \leftarrow \mathsf{GGen}(1^\lambda)$, $\mathbf{A} \leftarrow\!\$\, \mathcal{D}_{\ell,k}, \mathbf{w} \leftarrow\!\$\, \mathbb{Z}_q^k, [\mathbf{z}]_\gamma \leftarrow\!\$\, \mathbb{G}_\gamma^\ell$ and the coin tosses of adversary $\mathcal{A}$.

Theorem 3. *The split PKE* PKE *described in Fig. 9 is linearly key-homomorphic and lRCCA-secure w.r.t. f such that $f(\mathsf{sk}) \coloneqq \mathsf{sk}_B$ under the $\mathcal{D}_{k+1,k}$-MDDH assumption.*

Proof. To show that the scheme PKE of [FFHR19] is linear key-homomorphic, we briefly sketch that there exist the algorithms GenPK and $\mathsf{CheckPK}$ satisfyng the property. The KGen algorithm samples the secret material $\mathsf{sk} \coloneqq (\mathbf{a}, \mathbf{f}, \mathbf{F}, \mathbf{g}, \mathbf{G})$, consisting of vectors and matrices of elements in $\mathbb{Z}_q$. Then, in order to compute the public key pk, the secret key is projected: GenPK produces some matrices $([\tilde{\mathbf{f}}_j]_T, [\tilde{\mathbf{F}}_j]_1, [\tilde{\mathbf{g}}_j]_T, [\tilde{\mathbf{G}}_j]_2, [\tilde{\mathbf{H}}_j]_1, [\tilde{\mathbf{I}}_j]_2)$. It is also immediate to check that, for any $\alpha \in \mathbb{Z}_q$ and any two secret keys $\mathsf{sk}_1, \mathsf{sk}_2$, the linear homomorphic property holds, i.e. $\mathsf{GenPK}(\alpha \cdot \mathsf{sk}_1 + \mathsf{sk}_2) = \alpha \cdot \mathsf{GenPK}(\mathsf{sk}_1) + \mathsf{GenPK}(\mathsf{sk}_2)$. To verify that $\mathsf{CheckPK}(\mathsf{pk})$, instead, it is sufficient to check the following pairing equations:

$$e([\tilde{\mathbf{H}}]_1, [\mathbf{E}]_2) = e([\mathbf{D}^*]_1, [\tilde{\mathbf{G}}_j]_2)$$
$$e([\mathbf{D}]_1, [\tilde{\mathbf{I}}]_2) = e([\tilde{\mathbf{F}}]_1, [\mathbf{E}]_2)$$

We now prove that the scheme is lRCCA-secure w.r.t. the function f, where $f(\mathsf{sk}) \coloneqq \mathsf{sk}_B$. We briefly recall the proof strategy from [FFHR19]. We only highlight the main differences.

- The hybrid $\mathbf{H}_1$ computes the challenge ciphertext using the secret key. Namely $p^* \leftarrow \mathbf{a}^\top \mathbf{u}^* + \mathtt{M}_b$ and similarly the proofs are $\pi_1^* \leftarrow \mathbf{f}^\top \mathbf{Dr} + \mathbf{v}^\top \mathbf{F}^\top \mathbf{DrF}$ and $\pi_2^* \leftarrow \mathbf{g}^\top \mathbf{Es} + \mathbf{x}^\top \mathbf{G}^\top \mathbf{Es}$. This hybrid is equivalent to the lRCCA experiment.
- The hybrid $\mathbf{H}_2$ samples for the challenge ciphertext $\mathbf{u}^*$ and $\mathbf{v}^*$ uniformly at random from $\mathbb{Z}_q^{k+1}$. The hybrids $\mathbf{H}_1$ and $\mathbf{H}_2$ are computationally indistinguishable. This follows by applying the $\mathcal{D}_k$-MDDH Assumption on $[\mathbf{D}, \mathbf{u}^*]_1$ in $\mathbb{G}_1$ and $[\mathbf{E}, \mathbf{v}^*]_2$ in $\mathbb{G}_2$. Notice that the reduction can sample sk_2 itself and easily reveal sk_2 to the adversary.

From now on, we prove that each pair of consecutive hybrids is statistically close. In particular, this means that the hybrids (and, in principle, also the adversary) are allowed to run in unbounded time.

– The hybrid $\mathbf{H}_3$ adds the two decryption rules, if $\mathbf{u} = \mathbf{D}\mathbf{r}$ uses $\mathbf{r}$ to compute π_1 and $\mathtt{M}$ at decryption time, similarly, if $\mathbf{v} = \mathbf{E}\mathbf{s}$ then uses $\mathbf{r}$ to compute π_2. It is easy to see that this hybrid is equivalent to the previous by the linearity of the decryption procedure.
– The hybrid $\mathbf{H}_4$ adds another decryption rule, if $\mathbf{u} \notin \mathsf{Span}(\mathbf{D})$ and $\mathbf{v} - \mathbf{v}^* \notin \mathsf{Span}(\mathbf{E})$ then the decryption oracle returns $\perp$ to the adversary. This is the main core of the technique of [FFHR19]. In particular, they show that the probability that the proof π of the queried ciphertext is valid when the condition holds is $O(1/q)$. By inspection of their reduction, we notice that the reduction stops the simulation of the hybrid $\mathbf{H}_4$ as soon as a decryption query triggers the event in the newly added decryption rule. In particular, it means that the reduction does not have to simulate for the adversary the leakage value $f(\mathsf{sk}) = \mathsf{sk}_2$. In other words, the reduction [FFHR19] works exactly the same also in our leakage resilient experiment because it stops the adversary before the latter may receive the leaked value.
– The hybrid $\mathbf{H}_5$, similarly to the previous hybrid, adds a new the decryption rule that if $\mathbf{v} \notin \mathsf{Span}(\mathbf{E})$ and $\mathbf{x} - \mathbf{x}^* \notin \mathsf{Span}(\mathbf{D}^*)$ then the decryption oracle returns $\perp$ to the adversary. The proof for this hybrid is almost identical to the proof for the previous hybrid.
– The hybrid $\mathbf{H}_6$ adds a new decryption rule that iff $\mathbf{x} - \mathbf{x}^* \in \mathsf{Span}(\mathbf{D}^*)$ and $\mathbf{v} - \mathbf{v}^* \in \mathsf{Span}(\mathbf{E})$ then the proof is verified in an alternative way. In particular, let $\tilde{\mathbf{r}}, \tilde{\mathbf{s}}$ be such that $\mathbf{x} - \mathbf{x}^* = \tilde{\mathbf{x}} = \mathbf{D}\tilde{\mathbf{r}}$ it computes π' as a re-randomization of the proof of the challenge ciphertext π^* using randomness $\tilde{\mathbf{r}}$ and $\tilde{\mathbf{s}}$ and if $\pi' \neq \pi$ the decryption oracle returns $\perp$ to the adversary. The proof of this hybrid follows the correctness of the re-randomization algorithm for the PKE.
– The hybrid $\mathbf{H}_7$ is the same as the previous, but it never uses the secret key material sk to decrypt. This hybrid is only syntactically different than the previous; in fact, by the decryption rules added in the previous hybrids also the $\mathbf{H}_6$ would never use the secret key material for the decryption query.

At this point, we can show that $\mathbf{H}_6$ is independent of the challenge bit even if the adversary additionally gets to see the leakage sk_2. Notice that only dependence on b is given by $p^* = \mathbf{a}^\top \mathbf{u}^* + \mathtt{M}_b$. Moreover, b is independent of sk_2 thus, even leaking this value the view of the adversary is independent from the challenge bit.

5.4 Putting all together

We can instantiate the ABO Perfect Hiding NIZK proof of membership $\mathsf{NIZK}_{\mathsf{mx}}$ using Groth-Sahai proofs [EG14] . In particular, notice that the necessary tag-space for $\mathsf{NIZK}_{\mathsf{mx}}$ is the set $[m]$ which in typical scenarios is a constant small number (for example 3 mixers). Thus we can instantiate the tag-based ABO Perfect Hiding $\mathsf{NIZK}_{\mathsf{mx}}$ by considering an Init algorithm that samples m different common reference strings $(\mathsf{crs}_i)_{i\in[m]}$, the prover algorithm (resp. the verify algorithm) on tag j invokes the GS prover algorithm (resp. verifier algorithm) with input the common reference string crs_j. We can instantiate the tag-based ABO Perfect Sound NIZK $\mathsf{NIZK}_{\mathsf{sd}}$ using the technique presented in the full version of [FFHR19] (see Section 2.2 for more details). By the universal composability theorem [Can01], once we compose the protocol Π_{Mix} from Fig. 5 and Π_{VtDec} from Fig. 8 we obtain a protocol with setup assumption $\mathcal{F}_{\mathsf{Dec}}$, $\mathcal{F}_{\mathsf{Com}}$ and $\mathcal{F}_{\mathrm{CRS}}$. The first ideal functionality can be implemented using classical approaches (for example, see Benaloh [Ben06]). Briefly, the mixers can compute the shares of the public key $[\mathbf{a}^\top\mathbf{D}]_1$ for KGen_A as in Fig. 9 and prove the knowledge of the secret key share $\mathbf{a}^{(i)}$ where $\mathbf{a} = \sum_i \mathbf{a}^{(i)}$, to obtain UC security in the malicious setting against static corruptions we can use an ABO Perfect Hiding NIZK proof system for this step. At decryption time, the mixers can compute a batched zero-knowledge proof of knowledge for "*encryption of zero*", they can use a NIZK proof of membership and, for UC security, it is sufficient for such proofs to be adaptive perfect sound.

Auditability. Here we sketch the auditability of our protocol. Roughly speaking, a protocol Π is *auditable* if there exists a PT algorithm Audit that on input a transcript τ and an output y output 1 if and only if the execution of the protocol that produces the transcript τ ends up with the parties outputting y. We focus on the auditability of the protocol obtained composing Π_{Mix} from Fig. 5 and Π_{VtDec} from Fig. 8. The auditing algorithm, given a transcript of Π_{VtDec} can reconstruct the secret key sk_2 and can check that $\mathsf{Vf}(\mathsf{sk}_2, \mathsf{C}_{i,j}) = 1$ for all $i \in [m]$ and $j \in [n]$ moreover it checks that all the NIZK proofs verify. The checks performed guarantee that the protocol execution resulting to the transcript did not abort, moreover, the auditability is guaranteed by the correctness of the protocol. Finally, we notice that the protocol for $\mathcal{F}_{\mathsf{Dec}}$ sketched in the previous section is auditable (see [Ben06]).

Efficiency. We analyze the efficiency of the protocol obtained composing Π_{Mix} and Π_{VtDec}, and we consider the most efficient instantiation of the scheme in [FFHR19] based on SXDH assumption, i.e. for $k = 1$. We denote with E_1, E_2 (resp. E_T) the cost of a multiplication in groups $\mathbb{G}_1$ and $\mathbb{G}_2$ (resp. exponentiation in $\mathbb{G}_T$), and with P the cost of computing a bilinear pairing. We give an intuition on how much the protocol scales when a mixer is given N processors and may make use of parallelism. We compare our results with the Mix-Net protocol of [FFHR19]. In our protocol Π_{Mix}, each mixer re-randomizes a list of n ciphertexts which requires $n(7E_1+7E_2+2E_T+9P)$, and additionally computes a proof π_{mx} for the sumcheck relation $\mathcal{R}_{\mathsf{mx}}$ which requires n additions in $\mathbb{Z}_q$ and $6E_1 + 8E_2$. Re-randomization of a ciphertext in the list does not depend on other ciphertexts in the list, so the parallel cost is $\frac{n}{N}(7E_1 + 7E_2 + 2E_T + 9P)$. Additionally, the mixers verify all the sumcheck NIZK proofs, which requires $3nm$ additions in $\mathbb{G}_1$ and around 8 pairings. The parallel cost is $\frac{8m}{N}$ pairings plus $\log_N(3n)\frac{m}{N}$ additions.

In the protocol Π_{VtDec}, each mixer sends a commitment of their secret key share, which requires a UC-commitment for the elements of the secret key sk, and receives commitments of secret key shares of the other $m-1$ mixers. Additionally, the mixers derives the public key shares, using GenPK, this corresponds to the cost of generating m times a key pk_B^i and requires $m(4E_T + 6E_1 + 6E_2)$. Finally, each mixer needs to verify the $n \cdot m$ ciphertexts produced in the protocol execution of the last list which requires $n(m-1)(6E_1 + 4E_2 + 4P)$.

The protocol of [FFHR19] the public key shares pk_B^i (and not the secret ones) are committed using an equivocable commitment and an ABO NIZK proof (which can be seen as a UC-secure commitment against static corruption). The parallel cost of re-randomize their ciphertexts is $\frac{n}{N}36E_1 + 45E_2 + 6E_T + 5P$, while the cost of verifying the ciphertexts and decrypting the last list is equal to $\frac{nm}{N}36P + \frac{m}{N}(2E_1 + 50P)$. In comparison, our approach allows to save at least $\frac{n}{N}(30E_1 + 39E_2 + 36P)$ cryptographic operations, where we recall that n is the number of shuffled ciphertexts.

Acknowledgements

This work has been partially supported by the MESRI-BMBF French-German joint project named PROPOLIS (ANR-20-CYAL-0004-01).

References

Abe98. Masayuki Abe. Universally verifiable mix-net with verification work independent of the number of mix-servers. In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 437–447. Springer, Heidelberg, May / June 1998.

Abe99. Masayuki Abe. Mix-networks on permutation networks. In Kwok-Yan Lam, Eiji Okamoto, and Chaoping Xing, editors, *ASIACRYPT'99*, volume 1716 of *LNCS*, pages 258–273. Springer, Heidelberg, November 1999.

AH01. Masayuki Abe and Fumitaka Hoshino. Remarks on mix-network based on permutation networks. In Kwangjo Kim, editor, *PKC 2001*, volume 1992 of *LNCS*, pages 317–324. Springer, Heidelberg, February 2001.

AW07. Ben Adida and Douglas Wikström. Offline/online mixing. In Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, editors, *ICALP 2007*, volume 4596 of *LNCS*, pages 484–495. Springer, Heidelberg, July 2007.

Ben06. Josh Benaloh. Simple verifiable elections. In *2006 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT 06)*, Vancouver, B.C., August 2006. USENIX Association.

BG12. Stephanie Bayer and Jens Groth. Efficient zero-knowledge argument for correctness of a shuffle. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 263–280. Springer, Heidelberg, April 2012.

Can01. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

CF01. Ran Canetti and Marc Fischlin. Universally composable commitments. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 19–40. Springer, Heidelberg, August 2001.

Cha81. David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, February 1981.

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

CKN03. Ran Canetti, Hugo Krawczyk, and Jesper Buus Nielsen. Relaxing chosen-ciphertext security. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 565–582. Springer, Heidelberg, August 2003.

CS98. Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 13–25. Springer, Heidelberg, August 1998.

CS02. Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer, Heidelberg, April / May 2002.

EG14. Alex Escala and Jens Groth. Fine-tuning Groth-Sahai proofs. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 630–649. Springer, Heidelberg, March 2014.

EHK+13. Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Heidelberg, August 2013.

FF20. Antonio Faonio and Dario Fiore. Improving the efficiency of re-randomizable and replayable CCA secure public key encryption. In Mauro Conti, Jianying Zhou, Emiliano Casalicchio, and Angelo Spognardi, editors, *ACNS 20, Part I*, volume 12146 of *LNCS*, pages 271–291. Springer, Heidelberg, October 2020.

FFHR19. Antonio Faonio, Dario Fiore, Javier Herranz, and Carla Ràfols. Structure-preserving and re-randomizable RCCA-secure public key encryption and its applications. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 159–190. Springer, Heidelberg, December 2019.

FS01. Jun Furukawa and Kazue Sako. An efficient scheme for proving a shuffle. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 368–387. Springer, Heidelberg, August 2001.

GI08. Jens Groth and Yuval Ishai. Sub-linear zero-knowledge argument for correctness of a shuffle. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 379–396. Springer, Heidelberg, April 2008.

Gro03. Jens Groth. A verifiable secret shuffle of homomorphic encryptions. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 145–160. Springer, Heidelberg, January 2003.

Gro04. Jens Groth. Rerandomizable and replayable adaptive chosen ciphertext attack secure cryptosystems. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 152–170. Springer, Heidelberg, February 2004.

Gro10. Jens Groth. A verifiable secret shuffle of homomorphic encryptions. *Journal of Cryptology*, 23(4):546–579, October 2010.

JM99. Markus Jakobsson and David M'Raïhi. Mix-based electronic payments. In Stafford E. Tavares and Henk Meijer, editors, *SAC 1998*, volume 1556 of *LNCS*, pages 157–173. Springer, Heidelberg, August 1999.

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

LPQ17. Benoît Libert, Thomas Peters, and Chen Qian. Structure-preserving chosen-ciphertext security with shorter verifiable ciphertexts. In Serge Fehr, editor, *PKC 2017, Part I*, volume 10174 of *LNCS*, pages 247–276. Springer, Heidelberg, March 2017.

Nef01. C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In Michael K. Reiter and Pierangela Samarati, editors, *ACM CCS 2001*, pages 116–125. ACM Press, November 2001.

PIK94. Choonsik Park, Kazutomo Itoh, and Kaoru Kurosawa. Efficient anonymous channel and all/nothing election scheme. In Tor Helleseth, editor, *EUROCRYPT'93*, volume 765 of *LNCS*, pages 248–259. Springer, Heidelberg, May 1994.

PR07. Manoj Prabhakaran and Mike Rosulek. Rerandomizable RCCA encryption. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 517–534. Springer, Heidelberg, August 2007.

SK95. Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme - a practical solution to the implementation of a voting booth. In Louis C. Guillou and Jean-Jacques Quisquater, editors, *EUROCRYPT'95*, volume 921 of *LNCS*, pages 393–403. Springer, Heidelberg, May 1995.

TW10. Björn Terelius and Douglas Wikström. Proofs of restricted shuffles. In Daniel J. Bernstein and Tanja Lange, editors, *AFRICACRYPT 10*, volume 6055 of *LNCS*, pages 100–113. Springer, Heidelberg, May 2010.

WCY+21. Yi Wang, Rongmao Chen, Guomin Yang, Xinyi Huang, Baosheng Wang, and Moti Yung. Receiver-anonymity in rerandomizable RCCA-secure cryptosystems resolved. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 270–300, Virtual Event, August 2021. Springer, Heidelberg.

Wik05. Douglas Wikström. A sender verifiable mix-net and a new proof of a shuffle. In Bimal K. Roy, editor, *ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 273–292. Springer, Heidelberg, December 2005.

Wik09. Douglas Wikström. A commitment-consistent proof of a shuffle. In Colin Boyd and Juan Manuel González Nieto, editors, *ACISP 09*, volume 5594 of *LNCS*, pages 407–421. Springer, Heidelberg, July 2009.

Wik10. Douglas Wikström. Verificatum, 2010. https://www.verificatum.com.