Optimistic Mixing, Revisited

Antonio Faonio and Dario Fiore

IMDEA Software Institute, Madrid, Spain

Abstract. Mixing Networks are protocols that allow a set of senders to send messages anonymously. Such protocols are fundamental building blocks to achieve privacy in a variety of applications, such as anonymous e-mail, anonymous payments, and electronic voting. Back in 2002, Golle *et al.* proposed a new concept of mixing network, called optimistic mixing, that allows for fast mixing when all the parties execute the protocol honestly. If, on the other hand, one or more mix-servers cheat, then the attack is recognized and one can back up to a different, slow mix-net. Unfortunately, Abe and Imai (ACISP'03) and independently Wikström (SAC'03) showed several major flaws in the optimistic protocol of Golle *et al.* In this work, we give another look at optimistic mixing networks. Our contribution is mainly threefold. First, we give formal definitions for optimistic mixing in the UC model. Second, we propose a compiler for obtaining a UC-secure mixing network by combining an optimistic mixing with a traditional mixing protocol as backup mixing. Third, we propose an efficient UC-secure realization of optimistic mixing based on the DDH assumption in the non-programmable random oracle model. As a key ingredient of our construction, we give a new randomizable replayable-CCA secure public key encryption (PKE) that outperforms in efficiency all previous schemes. We believe this result is of independent interest.

Keywords: Mix-Nets, Re-Randomizable Replayable CCA, UC-security.

Table of Contents

1 Introdution

Bakground. Mixing Networks (aka mix-nets), originally proposed by Chaum [16], 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, if at least one of the permutations applied by the mixers remains secret, 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 used as fundamental building blocks to achieve privacy in a variety of application scenarios, including anonymous e-mail [16], anonymous payments [36], and electronic voting [16]. 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 a desired candidate!

Realizing Mix-Nets. A popular design paradigm of mixing networks are *re-encryption mix-nets* [41] 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 (or argument). This idea inspired a long series of works on zero-knowledge shuffle arguments, e.g., [24,40,31,51,53,34,48,9]. Notably, some recent works [53,48,9] improved significantly over the early solutions, and they have been implemented and tested in real-world applications (elections) [54]. 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 it is realistic for mix-nets to be able to scale up to millions of senders (i.e., voters).

Optimistic Mixing. Most of the research effort for improving efficiency of mix-nets has been so far devoted to improving the efficiency of shuffle arguments. A notable exception is the work of Golle *et al.* [30]. They proposed a new mixing network optimized to quickly produce a correct output when all the mix-servers execute the protocol correctly. If, on the other hand, one or more mix-servers cheat, then no privacy is guaranteed, but the attack is noticed, and one can decide to "back up" to a different, slow, mix-net execution. Their efficiency improvement came from an "optimistic" or "fast-track" [25] approach, from which the name *optimistic mixing*. Essentially, the idea is that optimistic mixing is not sufficient to realize a mix-net by itself, yet one can (optimistically) attempt an execution of the optimistic mix-net and, in the rare case of failure, "*backup*" to a slow mix-net. This way, one gets protocols that, in most realistic scenarios, run fast.

Unfortunately, one year after its proposal, Abe and Imai [5] and independently Wikström [49] showed major flaws in the optimistic protocol of Golle *et al.* The attacks leveraged malleability issues in the encryption scheme used in the protocol as well as a lack of a formal security definition for optimistic mixing.

1.1 Our Contribution

In this work we rescue the optimistic mixing approach. As main contributions, (1) we introduce formal definitions for optimistic mixing in the UC model, (2) we propose a compiler in the UC model for obtaining a secure mix-net through the composition of optimistic mixing with a traditional mixing protocol as backup mixing, and (3) we propose a UC-secure realization of optimistic mixing based on a new public key encryption (PKE) scheme that is re-randomizable and replayable-CCA (RCCA) secure (as defined by Canetti *et al.* [14]).

When instantiating our compiler with our new optimistic mix-net and the UC-secure mix-net of Wikström [51], we obtain a new UC-secure mixing network whose optimistic execution is faster than in existing UC-secure mix-nets [50,51]. In what follows we discuss our contributions in more detail.

Modeling Optimistic Mixing. One of the reasons that led to the break of Golle *et al.*'s optimistic mixing proposal was the lack of a clear security definition for this notion. To address this problem, our first contribution is to formally define the notion of an optimistic mix-net (OMix). We do this by using the framework of Universal Composability (UC) [12] that, in

addition to providing a way to clearly specify the security requirements of a protocol, offers the strong property that security is preserved even if the protocol is arbitrarily composed with itself or other protocols. We find the compositional property of UC particularly relevant to protocols like mix-nets that are naturally used in composition with larger cryptographic protocols.

For our definition, we start from the mix-net UC definition of Wikström [50] who models a mixing (Mix) protocol as one that realizes the functionality that collects inputs from the senders and outputs their lexicographic ordering, i.e., $\mathcal{F}^{\texttt{Mix}}(x_1, \ldots, x_n) = \mathsf{Sort}(x_1, \ldots, x_n)$. Inspired by this, we model an OMix network as a protocol that UC-securely realizes the functionality that either shuffles the input messages or does not, but in this case notifies all the parties that an attack was performed. More precisely,

$$\mathcal{F}^{\texttt{OMix}}(b, x_1, \ldots, x_n) = \begin{cases} (b, \mathsf{Sort}(x_1, \ldots, x_n)) & b = \texttt{valid} \\ (b, x_1, \ldots, x_n) & b = \texttt{invalid} \end{cases}$$

Above, b is a flag provided by the adversary. This means that the ideal adversary can decide whether the functionality provides a *layer of anonymization* of the inputs or not. While this functionality may seem useless if one wishes security against dishonest mixers, the original idea of Golle *et al.* [30] is that an OMix protocol needs to be backed up by another mixing protocol (let us call it the *backup mixing*) which would be run in case the ideal adversary decided to set b to `invalid`. In a sense, one aims to obtain $\mathcal{F}^{\texttt{Mix}}$ by composing $\mathcal{F}^{\texttt{OMix}} \circ \mathcal{F}^{\texttt{Mix}}$ where the composition consists in executing $\mathcal{F}^{\texttt{OMix}}$ first, and then, if the output contains $b = \texttt{invalid}$, executing $\mathcal{F}^{\texttt{Mix}}$ on the outptut of $\mathcal{F}^{\texttt{OMix}}$. The promising aspect of this composition approach is that valid executions only run the OMix protocol, and since this must realize a weaker functionality it might allow for faster realizations.

In our definitions we also consider the notion of public verifiability (aka auditability) for (optimistic) mixing. In a nutshell, auditability means that any third party can publicly verify that the output generated by running the mix-net on the private inputs is correct. To formalize this notion, we introduce a definitional framework for (UC composition of) auditable protocols with a bulletin board (BB). The idea is that the protocol's transcript is registered in the BB, and that an auditor can, in a non-interactive fashion, read this transcript and decide its correctness. This can be seen as an extension to the UC model of the auditable MPC notion of Baum *et al.* [8].

A 'Mix+OMix⇒Mix' Compiler. The above idea of composing an optimistic mixing with a traditional mixing requires care, though. For example, one challenge of the composition is that when the invalid branch is taken by the OMix protocol, the inputs cannot appear in clear before being sent to the backup mixing. The idea of Golle *et al.* to solve this problem, called double-enveloping, is that OMix runs on ciphertexts of the Mix protocol.

Our second contribution is a compiler that formalizes the double-enveloping idea (that in [30] was presented only informally), and fully clarifies the set of functional and security prerequisites that the backup mixing network must have to assure a secure composition. Specifically, we show that we can obtain a secure mix-net starting from an OMix network and what we call a *structured Mix* (sMix). The latter is simply any Mix protocol where senders submit their inputs encrypted (arguably the most common technique for input submission). In terms of security, we show that an sMix protocol must only realize a simpler, weaker functionality where there is a single sender submitting all the messages, and these messages are encrypted using a CCA2-secure encryption. We show that the UC-secure mix-net of Wikström [51] can be easily adapted to become structured, and thus can be used to instantiate our compiler. Hence, we actually give an 'sMix+OMix⇒Mix' compiler. This is stated and proven secure in the UC model, and therefore inherits the strong compositional properties of UC security.

Realizing Optimistic Mixing. We construct the first secure OMix network and prove it UC-secure in the non-programmable random-oracle (NPRO) model based on the DDH assumption. The first two main ingredients of our OMix protocol are: (1) a re-randomizable RCCA-secure public key encryption scheme, and (2) a non-interactive zero-knowledge (NIZK) proof of "loose shuffling" of ciphertexts of this PKE. We recall that RCCA security, introduced by Canetti, Krawczyk and Nielsen [14], is a relaxation of CCA2 security where one is allowed to maul a ciphertext if the underlying plaintext stays the same, as in the case of re-randomizations, whereas all other kinds of malleability would be detected.

In our OMix protocol, the senders use the RCCA PKE to encrypt their messages, and then each mixer re-randomizes the ciphertexts and creates a "loose shuffling" proof. For security, the key idea is that, despite this proof alone does not guarantee that the output ciphertexts are a shuffle, its combination with the RCCA property yields a shuffle proof. More in detail, if the "loose shuffling" is verified, the only other way to create a list that is not a shuffle would be to invalidate at least one ciphertext. So, if an invalid decryption occurs when decrypting the list returned by the last mixer, the OMix protocol raises an error.

This idea, however, does not work yet to produce a secure OMix protocol. The problem is that, to ensure that every mixer did a shuffle, one should decrypt the lists of ciphertexts returned by every mixer, but this would reveal their permutations. We solve this issue with our third key ingredient, which is (3) a novel technique called *authenticated re-randomization*. Informally, this offers the following guarantee: if an invalid ciphertext appears in a middle list, then it must propagate until the last list, or in other words it is infeasible for the adversary to repair invalid ciphertexts after they are re-randomized in an authenticated way. Authenticated re-randomization works by letting a mixer re-randomize using a secret-key (one for all the ciphertexts); this key performs a sort of homomorphic authentication on the ciphertexts and can be revealed after all mixers are done. Our proof shows that as long as there is one honest mixer performing this authentication we obtain the property that invalid ciphertexts cannot disappear.

Putting ideas (1), (2), and (3) together gives us a secure OMix protocol. To implement these ideas, our technical contributions are: a new PKE scheme that is re-randomizable, RCCA-secure, and has authenticated re-randomization; a NIZK proof of "loose shuffling" for the ciphertexts of this PKE. This NIZK is based on a simple sigma protocol for proving that a tuple of group elements lies in the span of a public vector. For n ciphertexts, this NIZK proof can be created and verified by doing $6n$ group multiplications and 5 exponentiations, which is extremely fast considering that exponentiations are way more expensive than multiplications. So, the work to be performed by every mixer is dominated by the authenticated re-randomization of ciphertexts, which depends on the efficiency of the PKE. Considering the overhead of double-enveloping, and using standard precomputation techniques, this requires 9.2 exponentiations per ciphertext (plus other 1.8 that can be precomputed offline prior to receiving the ciphertexts). Moreover, since re-randomization happens independently for every ciphertext, this operation can be highly parallelizable. These costs are comparable with those of state-of-the-art mixing protocols and are about 3 times faster than existing UC-secure mix-nets [50,51] (some details are in Sec. 7).

As a drawback, our protocol relies on a distributed decryption functionality that we have to instantiate with general-purpose techniques and whose practical realization is left as open problem. Nevertheless, we see our work as a first step in exploring (in a rigorous way) the potential of the optimistic approach, and we believe that further improvements can be achieved by investigating more the proposed paradigm. In particular, since the largest bulk of work depends from the efficiency of the RCCA scheme, any other new candidate performing better would yield more efficient optimistic mixing protocols.

PKE	Enc	Dec	Rand	$\lvert\mathsf{C}\rvert$	$\lvert\mathsf{pk}\rvert$	RCCA	rnd-RCCA	Ass.	Model
[32] Gro1	$3k+2$	$4k$	$3k+1$	$3k+1$	$3k+3$	weak	perfect	DDH	std
[32] Gro2	$3k+3$	$4k+3$	$3k+3$	$3k+2$	$3k+3$	full	perfect	–	GGM
[43] PR07	22	32	24	20	11	full	perfect	DDH*	std
This paper	16	18	15	11	11	full	weak	DDH	NPRO

Table 1. Comparison among a selection of re-randomizable and RCCA-secure PKE schemes. Encryption, decryption and randomization are measured in number of exponentiations; ciphertexts and public key in number of group elements. For the schemes of [32], k is the message bit-length, in the second scheme the group is $\mathbb{Z}^*_{N^2}$ for an RSA modulus N, GGM refers to the generic group model, and DDH* refers to the use of special groups. rnd-RCCA stands for re-randomizability in the presence of an RCCA decryption oracle. By "weak" RCCA (resp. rnd-RCCA) we mean the scheme satisfies the RCCA (resp. rnd-RCCA) notion in the presence of a weak-RCCA decryption oracle, whereas "full" means that the notion is satisfied with the standard RCCA decryption oracle (see Def. 10 and Fig. 4 for more details). Perfect re-randomizability, where the unbounded adversary gets the secret key, implies full rnd-RCCA.

A new re-randomizable RCCA-secure PKE. As we mentioned in the previous paragraph, a key ingredient of our OMix protocol realization is a new public key encryption that is RCCA-secure and re-randomizable. Our scheme is proven secure in the NPRO model under the DDH assumption and, to the best of our knowledge, outperforms all previous re-randomizable RCCA-secure schemes in all fronts. For fairness, we should mention that we achieve a weaker notion of re-randomizability that holds computationally and in which the adversary has access to a weaker RCCA oracle (as defined by Groth [32]) that outputs "invalid" (instead of "same") even when the decrypted message is the challenge one. We stress that this weaker oracle is used only for re-randomizability, whereas we achieve full-fledged RCCA security.

In Table 1 we present a comparison with other schemes.[1] We note that the scheme of Prabhakaran and Rosulek [43] works over two specific groups that are the quadratic residues subgroups of $\mathbb{Z}^*_{2q+1}$ and $\mathbb{Z}^*_{4q+3}$ respectively, where $(q, 2q+1, 4q+3)$ is a Cunningham chain of the first kind of length 3. In contrast, our scheme can be instantiated over *any group* where the DDH assumption holds, notably including elliptic-curve groups, which offer better performance.

1.2 Other related work

The notion of mix-net was introduced by Chaum [16]. The use of zero-knowledge arguments to prove the correctness of a shuffle was first suggested by Sako and Kilian [45]. The first proposals used expensive cut-and-choose based zero-knowledge techniques [45,2]. Abe *et al.* removed the need of cut-and-choose by proposing a shuffle based on permutation networks [3,4]. Furukawa and Sako [24] and independently Neff [40] 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 [53], and later Terelius and Wikström [48], who proposed arguments where the proof generation can be split into an offline and online phase (based on an idea of Adida and Wikström [6]). These protocols have been implemented in the Verificatum library [54]. Groth and Ishai [35] proposed the first zero-knowledge shuffle argument with sublinear communication. Bayer and Groth gave a faster argument with sublinear communication in [9]. A definition of mix-nets in the UC-framework was given by Wikström in [50], and another UC-secure realization appears in [51]. The concept of optimistic security is closely related to the concept of covert security [7]. On one hand, optimistic security specializes covert security, in the sense that the attacker is always caught when cheating; on the other hand, in our formulation optimistic secure protocols do not identify which party cheated.

[1] The table does not include the schemes in [15,37], which achieve the nice property that validity of ciphertexts can be checked publicly, but perform way worse than ours, e.g., a ciphertext contains 52 group elements.

2 Preliminaries

2.1 Basic Notation

For a binary string x, we denote respectively its length by $|x|$ and its i-th bit by x_i; if X is a set, $|X|$ represents the number of elements in X. When x is chosen randomly in X, we write $x \leftarrow_{\$} X$. When A is an algorithm, we write $y \leftarrow \mathsf{A}(x)$ to denote a run of A on input x and output y; if A is randomized, then y is a random variable and $\mathsf{A}(x;r)$ denotes a run of A on input x and randomness r. An algorithm A is *probabilistic polynomial-time* (PPT) if A is randomized and for any input $x, r \in \{0,1\}^*$ the computation of $\mathsf{A}(x;r)$ terminates in a polynomial number of steps (in the size of the input). We let $[n]$ be the set $\{1,\ldots,n\}$ and denote with $\mathcal{S}_n$ the group of permutations over $[n]$. In this paper we differentiate between *ordered* list of elements $\langle x_1,\ldots,x_n\rangle$, namely, list of elements where the positions of the elements $x_1,\ldots,x_n$ in the list is leaked, and not-ordered lists of elements (i.e., sets) $\{x_1,\ldots,x_n\}$ where the order does not matter.

Negligible functions. We denote with $\lambda \in \mathbb{N}$ the security parameter. A function $\nu : \mathbb{N} \to [0,1]$ is negligible in the security parameter (or simply negligible) if it vanishes faster than the inverse of any polynomial in λ, i.e. for any positive polynomial $p(\lambda)$ there exists a constant λ_0 such that for all $\lambda \geq \lambda_0$ we have $\nu(\lambda) \leq 1/p(\lambda)$. Equivalently, $\nu(\lambda) \in O(1/p(\lambda))$ for all positive polynomials $p(\lambda)$. We often write $\nu(\lambda) \in \mathtt{negl}(\lambda)$ to denote that $\nu(\lambda)$ is negligible.

Random variables and indistinguishability. For a random variable X, we write $\Pr[X = x]$ for the probability that X takes on a particular value $x \in \{X\}$ (with $\{X\}$ being the set where X is defined). The statistical distance between two random variables X and X' defined over the same set $\mathcal{X}$ is defined as $\mathbb{SD}(X;X') = \frac{1}{2}\sum_{x\in\{X\}} |\Pr[X=x] - \Pr[X'=x]|$.

Given two ensembles $X = \{X_\lambda\}_{\lambda\in\mathbb{N}}$ and $Y = \{Y_\lambda\}_{\lambda\in\mathbb{N}}$, we write: $X \approx_s Y$ to denote that the two ensembles are *statistically indistinguishable*, i.e., there exists a negligible function $\nu : \mathbb{N} \to [0,1]$ such that $\mathbb{SD}(X;Y) \leq \nu(\lambda)$; $X \equiv Y$ to denote that the two ensembles are *identically distributed*, i.e., that $\mathbb{SD}(X;Y) = 0$; and $X \approx_c Y$ to denote that they are *computationally indistinguishable*, i.e., for all PPT algorithms D there exists a negligible function $\nu : \mathbb{N} \to [0,1]$ such that $|\Pr[\mathsf{D}(X_\lambda) = 1] - \Pr[\mathsf{D}(Y_\lambda) = 1]| \leq \nu(\lambda)$.

Lemma 1 (Shoup's difference lemma [47]). *Let A, B, F be events defined in some probability distribution, and suppose that $A \wedge \neg F \Leftrightarrow B \wedge \neg F$. Then $|\Pr[A] - \Pr[B]| \leq \Pr[F]$.*

2.2 Non-Interactive Zero-Knowledge

A non-interactive zero-knowledge (NIZK) proof system for a relation $\mathcal{R}$ is a tuple $\mathcal{NIZK} = (\mathsf{Init}, \mathsf{P}, \mathsf{V})$ of PPT algorithms such that: Init on input the security parameter outputs a common reference string crs; $\mathsf{P}(\mathsf{crs}, x, w)$, given $(x,w) \in \mathcal{R}$, outputs a proof Π; $\mathsf{V}(\mathsf{crs}, x, \Pi)$, given instance x and proof Π outputs 0 (reject) or 1 (accept). In the protocols of this paper we use the notion of *NIZK with labels*, that are NIZKs where P and V additionally take as input a label L (e.g., a binary string). A NIZK (with labels) is *correct* if for every $\mathsf{crs} \leftarrow_{\$} \mathsf{Init}(1^\lambda)$, any label L, and any $(xw) \in \mathcal{R}$, we have $\mathsf{V}(\mathsf{crs}, L, x, \mathsf{P}(\mathsf{crs}, L, x, w)) = 1$.

We define three properties of a NIZK argument system.

Definition 1 (Adaptive soundness). *A NIZK with labels $\mathcal{NIZK}$ for relation $\mathcal{R}$ is adaptively sound if for every PPT adversary A, the following probability is negligible:*

$$\Pr\left[\begin{array}{c|c} (x,\Pi) \leftarrow \mathsf{A}(\mathsf{crs}), & \\ x \notin \mathcal{R} & \mathsf{crs} \leftarrow \mathsf{Init}(1^\lambda) \\ \mathsf{V}(\mathsf{crs}, x, \Pi) = 1 & \end{array}\right].$$

If soundness holds against unbounded adversaries we call $\mathcal{NIZK}$ a proof system, else it is an argument.

Definition 2 (Adaptive multi-theorem zero-knowledge, [22]). *A NIZK with labels $\mathcal{NIZK}$ for relation $\mathcal{R}$ satisfies adaptive multi-theorem zero-knowledge if the following holds:*

(i) There exists an algorithm $\overline{\mathsf{Init}}$ that outputs crs, and a simulation trapdoor td^s.
(ii) There exists a PPT simulator Sim such that, for all PPT adversaries A, we have that

$$\left|\Pr\left[\mathsf{A}^{\mathsf{P}(\mathsf{crs},\cdot,\cdot,\cdot)}(\mathsf{crs})=1|\ \mathsf{crs}\leftarrow\mathsf{Init}(1^\lambda)\right]-\Pr\left[\mathsf{A}^{\mathcal{SIM}(\cdot,\cdot)}(\mathsf{crs})=1|\ (\mathsf{crs},\mathsf{td}^s)\leftarrow\overline{\mathsf{Init}}(1^\lambda)\right]\right|$$

is negligible in λ. The simulation oracle $\mathcal{SIM}(\cdot,\cdot,\cdot)$ answers queries of the form (L,x,w) by checking if $(x,w)\in\mathcal{R}$, if so returning $\mathsf{Sim}(\mathsf{td}^s,L,x)$ otherwise returning $\perp$.

Groth [33] introduced the concept of simulation-extractable (SE) NIZK, which informally states that extractability holds even in presence of a simulator that simulates NIZK proofs for (possibly false) statements. In our paper we use a relaxed notion of simulation extractability where the extractor only returns a (possibly not invertible) function f of the witness. Specifically, in our case we use an f that outputs only a portion of the witness.

Definition 3 (simulation f-extractability). *A NIZK with labels $\mathcal{NIZK}$ for relation $\mathcal{R}$ is* f-simulation extractable *(f-SE) if the following holds:*

(i) There exists an algorithm $\overline{\mathsf{Init}}$ that outputs crs, a simulation trapdoor td^s, and an extraction trapdoor td^e.
(ii) There exists a PPT algorithm $\mathsf{Ext}(\mathsf{td}^e,(L,x),\Pi)$ such that every PPT adversary A has negligible probability of winning in the following game:
- *The challenger runs $(\mathsf{crs},\mathsf{td}^s,\mathsf{td}^e)\leftarrow\overline{\mathsf{Init}}(1^\lambda)$, and gives crs to A.*
- *A is given access to the simulation oracle $\mathcal{SIM}^*(\cdot,\cdot)$, which it can be queried on pairs of the form (L,x) of its choice, and $\mathcal{SIM}^*$ returns $\mathsf{Sim}(\mathsf{td}^s,L,x)$, where Sim is the simulator given by the zero-knowledge property.*
- *A outputs a tuple (L^*,x^*,Π^*).*
- *The challenger runs $y\leftarrow\mathsf{Ext}(\mathsf{td}^e,(L^*,x^*),\Pi^*)$.*

We say that A wins if: (a) (L^*,x^*) was not queried to the simulation oracle; (b) $\mathsf{V}(\mathsf{crs},(L^*,x^*),\Pi^*)=1$; (c) for any w such that $f(w)=y$, $(x^*,w)\notin\mathcal{R}$.

2.3 UC Security Model

We use the Universal Composability model of Canetti [12]. Specifically, our notation for multi-party protocols comes from the book of Cramer, Damgård, and Nielsen [17]. In this formulation of the UC model, the basic computational components are called *interactive agents*. Briefly, an interactive agent $\mathcal{A}$ is a computational device that receives and sends messages on named ports, and holds an internal state. More in particular, an interactive agent $\mathcal{A}$ has a collection of named *inports* where it receives messages and a collection of named *outports* where it sends messages. Given two agents $\mathcal{A}$ and $\mathcal{B}$ the union of them, denoted as $\mathcal{A}\circ\mathcal{B}$ is the *interactive system* where every outport of $\mathcal{A}$ (resp. $\mathcal{B}$) with name $\mathtt{N}$ is connected to an inport of $\mathcal{B}$ (resp. $\mathcal{A}$) with the same name $\mathtt{N}$ (if it exists). An interactive system where all the inports and outports are connected is said *closed*. The collection of all the ports that are not connected is called the *open ports* of the interactive system.

All the agents in the framework have one or more inports with name ending in $\mathtt{infl}$ or $\mathtt{infl}_i$. We call these special ports the *influence ports* of the agent. If an agent has a inport

with named N.infl then it must have a matching port with name N.lk. We call these special ports the *leakage ports* of the agent. Leakage and influence ports are very important: agents are activated by a special message on the influence port and they return their activation on the leakage port; parties of a protocol are corrupted by a special message on the influence port and from that point on they return their full state on their leakage port and proxy all the messages from the influence port (resp. to the leakage port) to (resp. from) their other ports (in this paper we consider only *active* corruption of the parties); the leakage and influence port of an ideal functionality model which information is allowed to leak and how an adversary can influence the functionality[2] (for more details see chapter 4 of [17]).

The agent of an ideal functionality with name F additionally has n inports named $\mathtt{F.in}_1, \ldots,$ $\mathtt{F.in}_n$, and n outports named $\mathtt{F.out}_1, \ldots, \mathtt{F.out}_n$. These $2n$ ports are called the *protocol ports*. A protocol Π consists of n parties $\mathcal{P}_1, \ldots, \mathcal{P}_n$ where each party is an interactive agent. A protocol Π has a *protocol name* F. We let a protocol Π and the ideal functionality $\mathcal{F}$ that Π is supposed to implement have the same name. We name the port N of the protocol Π as F.N. A protocol also has ports connecting it to the resource $\mathcal{R}$ with *resource name* R. The *resource* $\mathcal{R}$ takes care of moving messages between parties of the protocol. In addition to moving messages, it also models the leakage of the communication network and the possible influence that an attacker might have on the network.

A simulator S for a protocol Π with name F is defined as a polytime interactive system with an open inport named F.infl and an open outport named F.lk; these two ports allow S to connect to an ideal functionality $\mathcal{F}$ with name F. In addition to these two ports, a simulator has a collection of ports corresponding to all the open ports of the protocol Π with resource $\mathcal{R}$. As a consequence, the interactive systems $(\mathcal{F} \circ \mathsf{S})$ and $(\Pi \circ \mathcal{R})$ have the same open ports.

An environment $\mathcal{Z}$ for a protocol Π with resource $\mathcal{R}$ is an interactive system that has the dual open ports of $\Pi \circ \mathcal{R}$ (namely, $\mathcal{Z} \circ \Pi \circ \mathcal{R}$ forms a closed interactive system). In this paper we consider protocols that are secure against environments which performs static corruption of the parties. This means that corruption takes place before the protocol starts its execution and, as already mentioned above, the adversary can deviate from the protocol specification in any arbitrary way. Let Env^{static} be the set of all polytime interactive agents $\mathcal{Z}$ of this flavor. Specifically, $\mathcal{Z} \in Enc^{\mathsf{static}}$ if, at the first activation, and only in this activation, it sends a list of messages corrupt to the influence ports of the protocol. Two systems are indistinguishable to an environment $\mathcal{Z}$ if it cannot tell them apart, expect with negligible advantage, by sending and receiving messages on the open ports of the systems. More formally, the environment plays with a system and then, at the end of the execution outputs a bit b. Two systems $\mathcal{A}$ and $\mathcal{B}$ are indistinguishable to a class of environments Env if for any $\mathcal{Z} \in Env$ the bit output by $\mathcal{Z}$ interacting with $\mathcal{A}$ and the bit output by $\mathcal{Z}$ interacting with $\mathcal{B}$ are indistinguishable.

Definition 4. *Let* F *and* R *denote arbitrary protocol names. Let $\mathcal{F}$ be any ideal functionality with name* F*, Π be any protocol with protocol name* F *and resource name* R*, and let $\mathcal{R}$ be an ideal functionality with name* R*. We say that $\Pi \circ \mathcal{R}$* securely implements *$\mathcal{F}$ in the environments from Env if there exist a simulator S such that $\Pi \circ \mathcal{R}$ and $\mathsf{S} \circ \mathcal{F}$ are indistinguishable to the class Env.*

We are ready to state the UC Theorem (see Thm 4.20 of [17]).

Theorem 1 (The UC Theorem). *Let Env be a class of environments. Let $\Pi_{\mathtt{F}}$ a protocol with protocol name* F *and resource name* G*. Let $\Pi_{\mathtt{G}}$ a protocol with protocol name* G *and resource name* H*. Let $\mathcal{F}, \mathcal{G}, \mathcal{H}$ be ideal functionalities with name* F, G, H*.*

[2] For example, the ideal functionality of a private channel might send message on the leakage port of the form "P_1 sent a message to P_2 of lenght n" and might receive message on influence port of the form "deliver the message to P_2".

If $\Pi_{\mathtt{F}} \circ \mathcal{G}$ securely implements $\mathcal{F}$ and $\Pi_{\mathtt{G}} \circ \mathcal{H}$ securely implements $\mathcal{G}$, then $(\Pi_{\mathtt{F}} \circ \Pi_{\mathtt{G}}) \circ \mathcal{H}$ securely implements $\mathcal{F}$.

2.4 Algebraic Notation and Assumptions

In this section we present notation and useful definitions related to the algebraic tools used in our constructions. Let $\mathsf{Setup}(1^\lambda)$ be an algorithm that upon input λ produces parameters $\mathtt{prm} = (\mathbb{G}, q, G)$ describing a group $\mathbb{G}$ of prime order $q > 2^\lambda$, with generator G. We use additive notation for the group operation, and we denote group elements using the bracket notation introduced by Escala *et al.* in [20]. Namely, for a $y \in \mathbb{Z}_q$ we let $[y]$ be the element $y \cdot G$. We write elements in $\mathbb{G}$ with capital letters and elements in $\mathbb{Z}_q$ with lower case. We indicate vectors with boldface (e.g. $\mathbf{a}, \mathbf{b}, \dots$) and matrices with capital bold face (e.g. $\mathbf{A}, \mathbf{B}, \cdots$). We indicate vectors of elements in $\mathbb{G}$ with overlined capital letters (e.g. $\bar{A}, \bar{B}, \dots$).

We briefly recall the DDH assumption that we use in Sec. 5.

Definition 5 (DDH). *The Decisional Diffie-Hellman Assumption assumption holds for* Setup*, if for any* $\mathtt{prm} = (\mathbb{G}, q, G) \leftarrow_\$ \mathsf{Setup}(1^\lambda)$*, any* $a, b, c \leftarrow_\$ \mathbb{Z}_q$ *and any PPT* A

$$|\Pr[\mathsf{A}(\mathtt{prm}, [a, b, a \cdot b]) = 1] - \Pr[\mathsf{A}(\mathtt{prm}, [a, b, c]) = 1]| \leq \mathtt{negl}(\lambda).$$

Useful facts. To analyze the security of our construction we rely on the following lemma, and a simple corollary of it.

Lemma 2 (Adaptive smoothness). *For all* $\mathtt{prm} \leftarrow_\$ \mathsf{Setup}(1^\lambda)$*, any* $\mathbf{g} \in \mathbb{Z}_q^2$*,* $\mathbf{a} \in \mathbb{Z}_q^m$*, and any (unbounded) adversary* A*:*

$$\Pr_{\mathbf{A} \leftarrow_\$ \mathbb{Z}_q^{m \times 2}, \mathsf{A}} \left[\begin{array}{c} [\mathbf{w}, \mathbf{z}] \leftarrow \mathsf{A}(\mathbf{a}), \\ \mathbf{w} \notin \mathit{Span}(\mathbf{g}) \\ \mathbf{z} = \mathbf{A} \cdot \mathbf{w} \end{array} \middle| \; \mathbf{A} \cdot \mathbf{g} = \mathbf{a}, \right] \leq 1/q^m.$$

Proof. We show that for any $\mathbf{a} \in \mathbb{Z}_q^m$, any A and any assignment of the randomness of A that leads to $[\mathbf{w}, \mathbf{z}]$ such that the event in the probability above holds there is 1 possible assignments of $\mathbf{A} \in \mathbb{Z}_q^{m \times 2}$. In fact, notice that $\mathbf{A}$ is such that:

$$\begin{cases} \mathbf{A} \cdot \mathbf{g} = \mathbf{a} \\ \mathbf{A} \cdot \mathbf{w} = \mathbf{z} \end{cases}$$

This is a system of $2m$ linear and independent equations over $2m$ variables in $\mathbb{Z}_q$, which therefore admits only 1 solution. Notice that, conditioned on $\mathbf{a}$, the matrix $\mathbf{A}$ is sampled uniformly at random from a set of size q^m, therefore the probability that $\mathbf{A}$ is a solution for the system above is $1/q^m$.

Corollary 1 (Non-Adaptive Smoothness). *For all* $\mathtt{prm} \leftarrow_\$ \mathsf{Setup}(1^\lambda)$*, any* $\mathbf{g} \in \mathbb{Z}_q^2$*, we have*

$$(\mathtt{prm}, \mathbf{A} \cdot \mathbf{g}, \mathbf{w}, \mathbf{A} \cdot \mathbf{w}]) \equiv (\mathtt{prm}, \mathbf{A} \cdot \mathbf{g}, \mathbf{w}, \mathbf{u}])$$

where $\mathbf{A} \leftarrow_\$ \mathbb{Z}_q^{m \times 2}, \mathbf{w} \leftarrow_\$ \mathbb{Z}_q^2, \mathbf{u} \leftarrow_\$ \mathbb{Z}_q^m$.

The corollary follows from Lemma 2 by observing that here $\mathbf{w}$ is chosen randomly and independently of $\mathbf{a} = \mathbf{A} \cdot \mathbf{g}$.

The two properties stated above are (slight) variants of the smoothness properties of Smooth Projective Hash Functions (see, e.g., [18]), in the case of a construction for the DDH language.

Although for simplicity we do not use the SPHF abstraction in our PKE scheme, we briefly recall the corresponding construction below. Let $[\mathbf{g}] \leftarrow_{\$} \mathbb{G}^2$ be a vector that defines the language $L = Span([\mathbf{g}]) \subset \mathbb{G}^2$. For a positive integer $m \in \mathbb{N}$, $\mathbf{A} \leftarrow_{\$} \mathbb{Z}_q^{m\times 2}$ is the secret hashing key, and $[\mathbf{a}] = [\mathbf{A} \cdot \mathbf{g}]$ the public projective key. Then, on input the projection key $[\mathbf{a}]$, word $[\mathbf{w}]$ and witness w such that $\mathbf{w} = w \cdot \mathbf{g}$, the public hashing outputs $w \cdot [\mathbf{a}]$. Instead, on input the hashing key $\mathbf{A}$ and any word $\mathbf{w}$, the (private) hashing outputs $[\mathbf{A} \cdot \mathbf{w}]$.

3 Verifiable and Optimistic Mixing Definitions

In this section we give definitions for Verifiable and Optimistic Mixing protocols in the UC model. In our work we are interested in protocols that are *auditable* (or, in other words, publicly verifiable). Informally speaking, auditability means that a third party (the auditor) can publicly verify that the output generated by executing the protocol on private inputs is correct.

To formalize this notion, we introduce a definitional framework for auditable protocols with a bulletin board (BB, for short). The basic idea is that the protocol's transcript, including the output, is written in the BB, and that an auditor can, in a non-interactive fashion, read such transcript from the BB and decide its correctness. Auditability was considered by Baum, Damgård and Orlandi [8]. Our definition follows in spirit their definition as we consider an external auditor party that cannot be corrupted, acts *after* the execution of the protocol and interacts only with the bulletin board. The main difference is that our definition is in the UC model and, loosely speaking, achieves what they call *auditable correctness* by comparing two experiments (in an ideal-versus-real-world fashion), whereas their definition asks explicitly that the probability of an auditor to accept (resp. reject) an output y is overwhelming when the functionality would output (resp. would not output) y over the input provided.

3.1 Auditable Protocols with Bulletin Board

In our model all the protocols have, as resource, a bulletin board BB with name $\mathtt{BB}$. The functionality bulletin board is the same as defined in [50], and can be realized using an authenticated broadcast channel (c.f. [28,38]). This resource is shared by all the protocols. To formalize this, we slightly tweak the notion of composition of interactive agents. Specifically, given two protocols Π_1 and Π_2 both with resource BB, we denote the composition of $(\Pi_1 \circ \mathrm{BB})$ and $(\Pi_2 \circ \mathrm{BB})$ as $(\Pi_1 \circ \Pi_2 \circ \mathrm{BB})$, where the protocol ports of the resource BB in the composed system are the union of the protocol ports of BB in $\Pi_1 \circ \mathrm{BB}$ and $\Pi_2 \circ \mathrm{BB}$.

An auditable protocol is a tuple (Π, Audit), where Audit is a non-interactive PPT algorithm and Π is a multi-party protocol. To define auditability we consider an auditor party $\mathcal{P}_A$ defined as an agent with name $\mathtt{A}$ and parametrized by an algorithm Audit (see Fig. 1). More in detail, the party $\mathcal{P}_A$ cannot be corrupted, (i.e., it would reject all corruption messages sent to its influence port), and whenever activated it reads the full content of the BB, it selects a relevant portion τ (e.g., the messages of the specific protocol execution to audit), runs the audit algorithm $b \leftarrow \mathsf{Audit}(\tau)$, and returns b on its leakage port.

To make auditable protocols composable, and in particular to enable the composability of Audit algorithms, we define an ideal audit procedure for the output of an ideal functionality. Notice that the output of an ideal functionality is, by definition, correct as it is produced as the correct computation on the private inputs. Therefore, given an ideal functionality/resource $\mathcal{F}$ with name $\mathtt{F}$ we define the ideal audit $\mathsf{Audit}^*_{\mathtt{F}}$ that trivially accepts if it finds in the BB a specific output authenticated by $\mathcal{F}$.

However, we need to instruct the ideal functionality to write its output messages on the bulletin board. To get this, given an ideal functionality $\mathcal{F}$, we consider a wrapper $\mathcal{W}[\mathcal{F}]$ that

proxies all the messages of $\mathcal{F}$ and also forwards all the messages in the protocol's output ports to the bulletin board using a special port name. This way, the bulletin board can syntactically distinguish what event to register in the "ideal board" $\mathcal{D}^I$ and what to register in the "real board" $\mathcal{D}^R$. The auditor party $\mathcal{P}_A$ is the only agent that can get access to the ideal board $\mathcal{D}^I$. Formally, let $\mathcal{D}^R, \mathcal{D}^I, X$ as in the description of the auditor agent in Fig. 1, we let $\mathsf{Audit}^*_{\mathtt{F}}(\mathcal{D}^I, X)$ return true iff $\exists c : (c, \mathtt{F}, X) \in \mathcal{D}^I$.

Definition 6. *Given an ideal functionality $\mathcal{F}$ with name $\mathtt{F}$, a resource $\mathcal{R}$ with name $\mathtt{R}$, and a class of environments Env, we say that an auditable protocol (Π, Audit), where Audit is a non-interactive PPT algorithm, and Π is a protocol with protocol name $\mathtt{F}$ and resource name $\mathtt{R}$, is* secure *for $\mathcal{F}$ with resource $\mathcal{R}$ for environments in Env if:*

1. *The protocol Π with resources $\mathcal{W}[\mathcal{R}]$ and* BB securely implements $\mathcal{F} \circ$ BB *for all environments $\mathcal{Z} \in Env$.*
2. *Let S be the simulator of Π, the agent systems $(\mathcal{W}[\mathsf{S}] \circ \mathcal{W}[\mathcal{F}] \circ \mathrm{BB} \circ \mathcal{P}_A[\mathsf{Audit}])$ and $(\mathcal{W}[\mathsf{S}] \circ \mathcal{W}[\mathcal{F}] \circ \mathrm{BB} \circ \mathcal{P}_A[\mathsf{Audit}^*_{\mathtt{F}}])$ are indistinguishable for all environments $\mathcal{Z} \in Env$.*

3.2 Verifiable Mixing Schemes and Optimistic Mixing Schemes

A Verifiable Mixing scheme and a (Verifiable) Optimistic Mixing scheme are auditable protocols that involve two groups of parties: (i) the senders $\{\mathcal{P}_{S_j}\}_{j\in[n]}$, (ii) and the mixers $\{\mathcal{P}_{M_i}\}_{i\in[m]}$. To define the security of VMix and OMix schemes we describe corresponding ideal functionalities $\mathcal{F}^{\mathtt{Mix}}$ in Fig. 2 and $\mathcal{F}^{\mathtt{OMix}}$ in Fig. 3.

In brief, the mixing functionality $\mathcal{F}^{\mathtt{Mix}}$ waits for an input $\mathtt{M}_j$ from every sender $\mathcal{P}_{S_j}$, and after all senders have submitted their inputs it returns these inputs in sorted order (essentially hiding who sent which message). The optimistic mixing functionality $\mathcal{F}^{\mathtt{OMix}}$ is almost the same as $\mathcal{F}^{\mathtt{Mix}}$ except that the adversary can send a special message `invalidate`. Then, if `invalidate` was sent, $\mathcal{F}^{\mathtt{OMix}}$ returns the senders' inputs in the same order as they were submitted, otherwise it returns the inputs in sorted order as $\mathcal{F}^{\mathtt{Mix}}$ does. In addition to the list of messages, $\mathcal{F}^{\mathtt{OMix}}$ outputs $b \in \{\mathtt{valid}, \mathtt{invalid}\}$ to indicate if `invalidate` was sent or not.

We stress that for both $\mathcal{F}^{\mathtt{Mix}}$ and $\mathcal{F}^{\mathtt{OMix}}$ the mixers do not need to provide any input to the functionalities. This feature is particularly useful in the UC model. For example, in our Mix-Net protocol, there is no need to UC-extract the permutations used by the mixers.

Definition 7. *A VMix $\mathcal{VM} = (\Pi, \mathsf{Audit})$ is $\mathcal{R}$-secure iff $\mathcal{VM}$ is a secure auditable protocol for $\mathcal{F}^{Mix}$ with resource $\mathcal{R}$ for environments in Env^{static}.*

Definition 8. *An OMix $\mathcal{OM} = (\Pi, \mathsf{Audit})$ is $\mathcal{R}$-optimistic-secure iff $\mathcal{OM}$ is a secure auditable protocol for $\mathcal{F}^{OMix}$ with resource $\mathcal{R}$ for environments in Env^{static}.*

4 RCCA PKE with Authenticated Re-randomizability

4.1 Re-randomizable RCCA PKE

A re-randomizable PKE scheme $\mathcal{PKE}$ is a tuple of five algorithms specified as follows.

$\mathsf{Setup}(1^\lambda)$ takes as input the security parameter λ (in unary) and produces public parameters $\mathtt{prm}$, which include a description of the message space $\mathcal{M}$.

$\mathsf{KGen}(\mathtt{prm}, 1^\ell)$ is the key generation algorithm that, on input the parameters $\mathtt{prm}$, outputs a key pair $(\mathsf{pk}, \mathsf{sk})$.

Wrapper $\mathcal{W}[\mathcal{F}]$ (resp. $\mathcal{W}[\mathsf{S}]$):

The wrapper has the same open ports of the functionality $\mathcal{F}$ with name $\mathtt{F}$ (resp. the simulator S with name $\mathtt{S}$). Moreover the wrapper has an outport with name $\mathtt{in}_{\mathtt{F}}$ that connects it to the bulletin board. (Resp. the wrapper for any ideal functionality simulated by S with name $\mathtt{F}$ as an outport with name $\mathtt{in}_{\mathtt{F}}$ that connects it to the bulletin board.)

1. On activation proxy all the messages on its inports to the inports of $\mathcal{F}$ and activate $\mathcal{F}$ (resp. S and activate S);
2. Once $\mathcal{F}$ (resp. S) returns its activation proxy all the messages on the outports of $\mathcal{F}$ (resp. S) to its outports, moreover if $\mathcal{F}$ has sent a message X on an protocol port $\mathtt{out}_i$ then send the message $(\mathtt{write}, \mathtt{F}, (i, X))$ to the port $\mathtt{in}_{\mathtt{F}}$ (resp. if S has sent a message X on a protocol port $\mathtt{F}, \mathtt{out}_i$ then send the message $(\mathtt{write}, \mathtt{F}, (i, X))$ to the port $\mathtt{in}_{\mathtt{F}}$).

Ideal functionality BB:

The ideal functionality has enough protocol ports to connect with all the protocols' agents of the agent system, for each ideal functionality in the agent system, it has special protocol ports named with $\mathtt{in}_{\mathtt{F}}$ where $\mathtt{F}$ is the name of the functionality, it has a special protocol inport $\mathtt{in}_{\mathtt{A}}$ and special protocol outport named $\mathtt{out}_{\mathtt{A}}$ (connected to the auditor party).

1. The functionality holds three databases $\mathcal{D}, \mathcal{D}^I$ and $\mathcal{D}^T$. We call $\mathcal{D}^I$ the database of the ideal functionalies' messages and $\mathcal{D}^T$ the database of the messages *on transit*. At first activation initialize the databases as empty, and the counters $c \leftarrow 1, c^I \leftarrow 1$.
2. Upon message $(\mathtt{write}, \mathsf{pid}, x)$ on the inport $\mathtt{BB.in}_i$, with $x \in \{0,1\}^*$, write (pid, i, x) on the database $\mathcal{D}^T$. Send the message $(\mathtt{write}, \mathsf{pid}, i, x)$ to the outport $\mathtt{BB.lk}$.
3. Upon message $(\mathtt{write}, X)$ on the inport $\mathtt{BB.infl}$, if X appears in $\mathcal{D}^T$ then write (c, X) on the database $\mathcal{D}$, delete the entry from $\mathcal{D}^T$, and increase the counter c.
4. Upon message $(\mathtt{read}, c)$ on the inport $\mathtt{BB.in}_i$ send the message $(\mathtt{read}, i, c)$ to the outport $\mathtt{BB.lk}$.
5. Upon message $(\mathtt{read}, i, c)$ on the inport $\mathtt{BB.infl}$, read from the database $\mathcal{D}$ the tuple (p, i, x) and if it exists write $(\mathtt{read}, (p, i, x))$ to the outport $\mathtt{BB.out}_i$ and to $\mathtt{BB.lk}$.

Special commands:

6. Upon message $(\mathtt{write}, \mathtt{F}, X)$ on the inport $\mathtt{BB.in}_{\mathtt{F}}$ write (c, X) on the database $\mathcal{D}^I$ and increase the counter c^I.
7. Upon message $\mathtt{read}$ on the inport $\mathtt{BB.in}_{\mathtt{A}}$, send the message $(\mathtt{read}, \mathcal{D}, \mathcal{D}^I)$ to the outport $\mathtt{BB.out}_{\mathtt{A}}$.

Agent $\mathcal{P}_A[\mathsf{Audit}]$:

The agent has two port $\mathtt{BB.in}_{\mathtt{A}}$ and $\mathtt{BB.out}_{\mathtt{A}}$ connected to the BB. Moreover, a protocol identifier pid is assigned to the agent.

1. Upon activation, if the message $\mathtt{corrupt}$ appears in the influence port then ignore it;
2. Read the message $(\mathtt{input}, X)$ from the influence port, send the message $\mathtt{read}$ to the outport $\mathtt{BB.in}_{\mathtt{A}}$ and at next activation read the message $(\mathtt{read}, \mathcal{D}, \mathcal{D}^I)$ from $\mathtt{BB.out}_{\mathtt{A}}$. Let $\mathcal{D}^R$ be the set $\{(c, i, x) : p = \mathsf{pid}, (c, p, i, x) \in \mathcal{D}\}$ and let $\tau \leftarrow (\mathcal{D}^R, \mathcal{D}^I)$.
3. If for all $i \in [n]$ there exists $(*, i, \mathtt{endProtocol})$ then compute $b \leftarrow \mathsf{Audit}(\tau, X)$ else return the message $\mathtt{error}$ to the port port $\mathtt{A.lk}$ and return;
4. Return the message $(\mathtt{audit}, b)$ to the port $\mathtt{A.lk}$.

Fig. 1: The wrapper for auditable ideal functionalities, the ideal functionality for Bulletin Board, and the auditor party.

Functionality $\mathcal{F}^{\mathtt{Mix}}$:

The functionality is implicitly parametrized by the security parameter 1^λ and has $n = \mathtt{poly}(\lambda)$ protocol ports with indices $\{S_j\}_{j\in[n]}$ and $m = \mathtt{poly}(\lambda)$ protocol ports with indices $\{M_i\}_{i\in[m]}$.

Init: At first activation read from $\mathtt{Mix.infl}$ the sets of corrupted players $\mathbf{C}^S, \mathbf{C}^M$, and set $\mathcal{I}^S \leftarrow \emptyset$, $\mathsf{flg}_{\mathtt{inpDone}} \leftarrow \mathsf{false}$, and $\mathsf{flg}_{\mathtt{mixDone}} \leftarrow \mathsf{false}$;

Input: On message $(\mathtt{input}, i, \mathtt{M}_i)$ on the inport $\mathtt{Mix.in}_{S_i}$ (or on the inport $\mathtt{Mix.infl}$ if $i \in \mathbf{C}^S$), if the entry $(i, *)$ does not exist in the database of the inputs then set $\mathcal{I}^S \leftarrow \mathcal{I}^S \cup \{i\}$, register the entry $(i, \mathtt{M}_i)$ in the database of the inputs, and send $(\mathtt{input}, i)$ on outport $\mathtt{Mix.lk}$.
If $|\mathcal{I}^S| = n$ then set $\mathsf{flg}_{\mathtt{inpDone}} \leftarrow \mathsf{true}$ and return.

Mix: On message $\mathtt{mixDone}$ on the inport $\mathtt{Mix.infl}$, if $\mathsf{flg}_{\mathtt{inpDone}} = \mathsf{true}$, compute $\mathcal{O} \leftarrow \mathsf{Sort}(\langle \mathtt{M}_j \rangle_{j\in[n]})$, send message $(\mathtt{mixDone}, \mathcal{O})$ on the outport $\mathtt{Mix.lk}$, and set $\mathsf{flg}_{\mathtt{mixDone}} = \mathsf{true}$.

Delivery: On message $(\mathtt{mixDone}, i)$, if $\mathsf{flg}_{\mathtt{mixDone}} = \mathsf{true}$, then send the message $(\mathtt{mixDone}, \mathcal{O})$ on the outport $\mathtt{Mix.out}_{M_i}$.

Fig. 2: Ideal Functionality for Mixing.

Functionality $\mathcal{F}^{\mathtt{OMix}}$:

The functionality is implicitly parametrized by the security parameter 1^λ, has name $\mathtt{OMix}$, and has $n = \mathtt{poly}(\lambda)$ protocol ports with indices $\{S_j\}_{j\in[n]}$ and $m = \mathtt{poly}(\lambda)$ protocol ports with indices $\{M_i\}_{i\in[m]}$.

Init: At first activation read from $\mathtt{OMix.infl}$ the sets of corrupted players $\mathbf{C}^S, \mathbf{C}^M$, and set $\mathcal{I}^S \leftarrow \emptyset$, $\mathsf{flg}_{\mathtt{inpDone}} \leftarrow \mathsf{false}$ and $\mathsf{flg}_{\mathtt{mixDone}} \leftarrow \mathsf{false}$, $b \leftarrow \mathtt{valid}$;

Input: On message $(\mathtt{input}, i, \mathtt{M}_i)$ on the inport $\mathtt{OMix.in}_{S_i}$ (or on the inport $\mathtt{OMix.infl}$ if $i \in \mathbf{C}^S$), if the entry $(i, *)$ does not exist in the database of the inputs, then set $\mathcal{I}^S \leftarrow \mathcal{I}^S \cup \{i\}$, register the entry $(i, \mathtt{M}_i)$ in the database of the inputs, and send $(\mathtt{input}, i)$ on outport $\mathtt{OMix.lk}$.
If $|\mathcal{I}^S| = n$ then set $\mathsf{flg}_{\mathtt{inpDone}} \leftarrow \mathsf{true}$ and return.

Mix: On message $\mathtt{mixDone}$ on the inport $\mathtt{OMix.infl}$, if $\mathsf{flg}_{\mathtt{inpDone}} = \mathsf{true}$:
- If $b = \mathtt{valid}$ then compute $\mathcal{O} \leftarrow \mathsf{Sort}(\langle \mathtt{M}_j \rangle_{j\in[n]})$;
- If $b = \mathtt{invalid}$ then compute $\mathcal{O} \leftarrow \langle \mathtt{M}_j \rangle_{j\in[n]}$.

send $(\mathtt{mixDone}, b, \mathcal{O})$ on the outport $\mathtt{OMix.lk}$ and set $\mathsf{flg}_{\mathtt{mixDone}} \leftarrow \mathsf{true}$.

Delivery: On message $(\mathtt{mixDone}, i)$, if $\mathsf{flg}_{\mathtt{mixDone}} = \mathsf{true}$, then send the message $(\mathtt{mixDone}, b, \mathcal{O})$ on the outport $\mathtt{OMix.out}_{M_i}$.

Invalidate: On message $\mathtt{invalidate}$ on the inport $\mathtt{OMix.infl}$ then set $b \leftarrow \mathtt{invalid}$ (resp. ignore the message).

Fig. 3: Ideal Functionality for Optimistic Mixing.

$\mathsf{Enc}(\mathsf{pk}, \mathtt{M})$ is the encryption algorithm that, on input a public key pk and a message $\mathtt{M} \in \mathcal{M}$, outputs a ciphertext $\mathtt{C}$;

$\mathsf{Dec}(\mathsf{sk}, \mathtt{C})$ is the decryption algorithm that, on input the secret key sk and a ciphertext $\mathtt{C}$, outputs a message $\mathtt{M} \in \mathcal{M}$ or an error symbol $\perp$.

$\mathsf{Rand}(\mathsf{pk}, \mathtt{C})$ is the randomization algorithm that, on input a public key pk and a ciphertext $\mathtt{C}$, outputs another ciphertext $\mathtt{C}'$;

We require the natural correctness property that for any pair $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{KGen}$ any randomization of a valid ciphertext under pk decrypts to the intended plaintext under sk.

We recall the notion of RCCA-PKE Security [14]. Very intuitively this can be thought of as a relaxation of the standard CCA security notion that allows for re-randomization of ciphertexts. A bit more technically, this is formalized with a security experiment that proceeds the same as the CCA security one except that in RCCA the decryption oracle can be queried on any ciphertext and, when decryption leads to one of the challenge messages $\mathtt{M}_0, \mathtt{M}_1$, it answers with a special symbol $\diamond$ (meaning "same").

Definition 9 (Replayable CCA Security.). *Consider the experiment* $\mathbf{Exp}^{\mathtt{RCCA}}$ *in Fig. 4, parametrized by a security parameter* λ*, an adversary* A*, and a PKE scheme* $\mathcal{PKE}$*. We say that*

$\mathcal{PKE}$ *is* indistinguishable secure under replayable chosen-ciphertext attacks *(RCCA-secure, for short) if there exists a negligible function* $\mathtt{negl}$ *such that for any PPT adversary* A

$$\left|\Pr\left[\mathbf{Exp}^{\mathtt{RCCA}}_{\mathsf{A},\mathcal{PKE}}(\lambda)=1\right]-\frac{1}{2}\right|\leq \mathtt{negl}(\lambda).$$

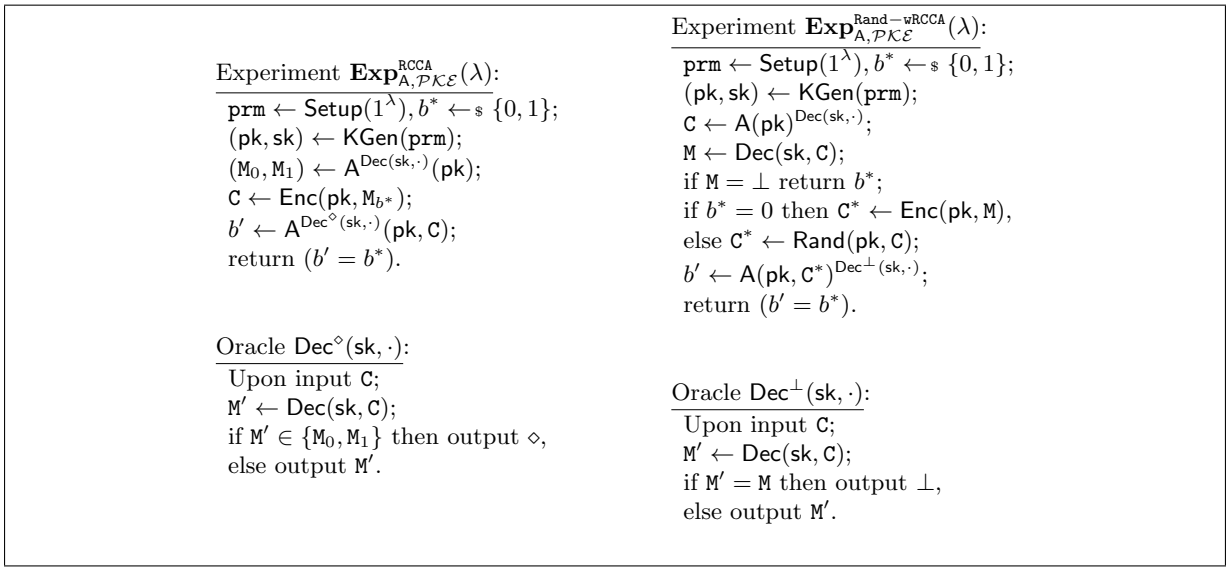

Fig. 4: The Re-Randomizable RCCA Security Experiments.

Second, we recall the notion of re-reradomizability for PKE. Intuitively, this notion asks that an adversary cannot tell apart a randomized ciphertext from a fresh new ciphertext for the same message. The strongest notion of re-randomizability (as considered in Groth [32] and Prabhajaran and Rosulek [43]), indeed, asks that the two distributions are identical, even conditioned on the knowledge of the secret material. In our work, we settle down for a weaker notion that considers indistinguishability for computationally bounded adversaries, which can still do a form of chosen-ciphertext attacks.[3] Specifically, we consider a decryption oracle that outputs the error message ($\perp$) either when the ciphertext does not decrypt or when it properly decrypts to the challenge message. This weaker oracle was considered by Groth [32] as a weakening of the RCCA one.

Definition 10 (Weak-RCCA Re-randomizability.). *Consider the experiment* $\mathbf{Exp}^{\mathtt{Rand-wRCCA}}$ *in Fig. 4. Let* $\mathcal{PKE}$ *be a re-randomizable PKE scheme.* $\mathcal{PKE}$ *is* rerandomizable under weak replayable chosen-chipertexts attacks *(Rand-wRCCA secure) if there is a negligible function* $\mathtt{negl}$ *such that for any PPT adversary* A

$$\left|\Pr\left[\mathbf{Exp}^{\mathtt{Rand-wRCCA}}_{\mathsf{A},\mathcal{PKE}}(\lambda)=1\right]-\frac{1}{2}\right|\leq \mathtt{negl}(\lambda).$$

4.2 Authenticated Re-randomizability

We introduce the notion of *(key-homomorphic) authenticated re-randomizability* for PKE schemes. Informally speaking, this notion ensures that after honest re-randomization, an adversarially

[3] Notice that the former notion captures chosen-ciphertext attacks thanks to the knowledge of the secret material.

generated ciphertext (which may be invalid), cannot be "repaired" to become valid. This property is crucial to counter "invalidate-and-repair" ciphertext tagging attacks that work as follows. Assume a mixing service consisting of three mixers such that each of them outputs a permuted re-randomization of the list of ciphertexts taken as input; also, assume that mixing is considered valid if all ciphertexts decrypt correctly (otherwise, an error is raised). Now, consider an adversary that controls the first and last mixer and, despite the honest behavior of the second mixer, wants to track a given input ciphertext, i.e., it wants to learn if the i-th ciphertext $\mathtt{C}_i$ given to the first mixer corresponds to the j-th ciphertext $\mathtt{C}'_j$ returned by the second mixer, thus de-anonymizing the i-th sender. The attack is the following: the first mixer invalidates $\mathtt{C}_i$ (replacing it with $\tilde{\mathtt{C}}_i$); the third mixer makes a corresponding reverse operation on $\mathtt{C}'_j$ (replacing it with $\tilde{\mathtt{C}}'_j$). Such operation has the property that, if $\mathtt{C}'_j$ is a re-randomization of $\tilde{\mathtt{C}}_i$, then $\tilde{\mathtt{C}}'_j$ is a *valid* re-randomization of $\mathtt{C}_i$. This way, if an error is *not* raised while decrypting $\tilde{\mathtt{C}}'_j$, the adversary learns that, with high probability, i was mapped into j. Notably, such an attack is not prevented by the RCCA property, and in fact it is possible to show RCCA schemes (e.g., the one in [43]) susceptible to it.

To solve this problem and make such repair infeasible, we introduce our new notion of authenticated re-randomizability. In terms of functionality, we require the randomization algorithm to use a authentication secret key κ, which must then be known by the decryption algorithm. Additionally, to let different independent parties re-randomize a ciphertext in a cascade fashion, we allow each party to use a different secret key, and decryption to work with the product of all these keys. Namely, $\mathsf{Rand}(\kappa_2, \mathsf{Rand}(\kappa_1, \mathtt{C}))$ can be decrypted using $(\kappa_1 \cdot \kappa_2)$ (this property is useful for efficiency, but not necessary for correctness). For security, we formalize a notion of unforgeability, which at a very high level says that an adversary who receives a ciphertext $\mathtt{C}$ re-randomized under secret key κ can only produce a ciphertext $\mathtt{C}'$ that is either invalid or decrypts (under κ) to the same value as $\mathtt{C}$, i.e., $\mathsf{Dec}(\mathsf{sk}, \kappa, \mathtt{C}) = \mathsf{Dec}(\mathsf{sk}, \kappa, \mathtt{C}')$, or $\mathsf{Dec}(\mathsf{sk}, \kappa, \mathtt{C}) = \perp$.

Formally, a re-randomizable PKE scheme $\mathcal{PKE} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Rand})$ has authenticated re-randomizability if it admits two PPT algorithms $(\mathsf{ADec}, \mathsf{ARand})$ and a group $\mathcal{K}$ acting as key space, that work as follows:

$\mathsf{ARand}(\mathsf{pk}, \kappa, \mathtt{C})$ is the authenticated re-randomization algorithm that takes as input a public key pk, a secret randomization key $\kappa \in \mathcal{K}$ and a ciphertext $\mathtt{C}$, and outputs a ciphertext $\mathtt{C}'$.

$\mathsf{ADec}(\mathsf{sk}, \kappa, \mathtt{C})$ is the authenticated decryption algorithm that takes as input the secret key sk, a randomization key $\kappa \in \mathcal{K}$ and a ciphertext $\mathtt{C}$, and outputs a message $\mathtt{M}$ or a special symbol $\perp$ (indicating an error).

For correctness, the following two properties must be satisfied.[4] For any $(\mathsf{sk}, \mathsf{pk}) \in \mathsf{KGen}(\mathtt{prm})$ and for any message $\mathtt{M}$: (1) For any $\kappa \in \mathcal{K}$ we have $\mathsf{ADec}(\mathsf{sk}, 1, \mathsf{Enc}(\mathsf{pk}, \mathtt{M})) = \mathtt{M}$; (2) For any $\kappa, \kappa' \in \mathcal{K}$ and for any $\mathtt{C}$ such that $\mathsf{Dec}(\mathsf{sk}, \kappa, \mathtt{C}) = \mathtt{M}$, we have $\mathsf{ADec}(\mathsf{sk}, \kappa \cdot \kappa', \mathsf{ARand}(\mathsf{pk}, \kappa', \mathtt{C})) = \mathtt{M}$.

We introduce the following security property.

Definition 11 (Unforgeability under Chosen-Chipertext Attacks). *Consider the experiment* $\mathbf{Exp}^{\mathtt{UF-CCA}}$ *in Fig. 5. Let* $\mathcal{APKE} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{ADec}, \mathsf{ARand})$ *be a PKE with authenticated re-randomizability. We say that* $\mathcal{APKE}$ *is* unforgeable under chosen-ciphertext attacks *(UF-CCA-secure, for short) if there exists a negligible function* $\mathtt{negl}$ *such that for any PPT* A

$$\Pr\left[\mathbf{Exp}^{\mathtt{UF-CCA}}_{\mathsf{A},\mathcal{APKE}}(\lambda) = 1\right] \leq \mathtt{negl}(\lambda).$$

[4] Notice that these properties generalize the correctness of a re-randomizable PKE.

Experiment $\mathbf{Exp}^{\mathtt{UF-CCA}}_{\mathsf{A},\mathcal{APKE}}(\lambda)$:
$\mathtt{prm} \leftarrow \mathsf{Setup}(1^\lambda)$,
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(\mathtt{prm}), \kappa \leftarrow\!\!\$\ \mathcal{K}$;
$(\{\mathtt{C}_i\}_{i\in[n]}, \tilde{\kappa}) \leftarrow \mathsf{A}(\mathsf{pk})$;
$\mathcal{C} = \{\mathtt{C}'_i \leftarrow \mathsf{ARand}(\mathsf{pk}, \kappa, \mathtt{C}_i)\}_{i\in[n]}$; $\mathcal{Q} = \{\mathsf{ADec}(\mathsf{sk}, \tilde{\kappa}, \mathtt{C}_i)\}_{i\in[n]}$;
$\mathtt{C}^* \leftarrow \mathsf{A}(\mathcal{C})$; $\mathtt{M}^* \leftarrow \mathsf{ADec}(\mathsf{sk}, \tilde{\kappa} \cdot \kappa, \mathtt{C}^*))$;
return 1 iff $(m^* \notin \mathcal{Q} \cup \{\perp\})$.

Fig. 5: UF-CCA security experiment.

Finally, for a PKE scheme with authenticated re-randomizability, the properties of RCCA security and weak-RCCA re-randomizability are required to hold with respect to the decryption oracle $\mathsf{ADec}(\mathsf{sk}, \cdot, \cdot)$ that takes inputs $(\kappa, \mathtt{C})$.

4.3 A Scheme Secure under DDH in the Random Oracle Model

We present our PKE scheme with authenticated re-randomizability. In order, we first present a re-randomizable PKE and prove that it is RCCA-secure and re-randomizable under the DDH assumption in the non-programmable random oracle model. Next, we show that the scheme can be extended to satisfy authenticated re-randomizability.

Our re-randomizable RCCA PKE. We are ready to describe the main construction of this section. The scheme works over a group $\mathbb{G}$ where the DDH problem is assumed intractable, and it supports the encryption of messages that are tuples of $\ell \in \mathbb{N}$ group elements. By abusing of the notation we let the key generation algorithm takes, as additional input, the length parameter $\ell = \mathtt{poly}(\lambda)$. The scheme $\mathcal{PKE} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Rand})$ is defined by the following tuple of algorithms.

- $\underline{\mathsf{Setup}(1^\lambda)}$: Choose a group $\mathbb{G}$ of prime order q such that $q > 2^\lambda$, and let G be a generator of $\mathbb{G}$. Output the group description $\mathtt{prm} = (\mathbb{G}, q, G)$.
- $\underline{\mathsf{KGen}(\mathtt{prm}, 1^\ell)}$: Sample $\mathbf{g} \leftarrow\!\!\$\ \mathbb{Z}_q^2$, $\mathbf{a}, \mathbf{c}, \mathbf{d} \leftarrow\!\!\$\ \mathbb{Z}_q^2$, $\mathbf{B} \leftarrow\!\!\$\ \mathbb{Z}_q^{\ell\times 2}$ and $\mathbf{F} \leftarrow\!\!\$\ \mathbb{Z}_q^{(\ell+3)\times 2}$. Compute $a \leftarrow \mathbf{a}^T \cdot \mathbf{g}$, $\mathbf{b} \leftarrow \mathbf{B} \cdot \mathbf{g}$, $c \leftarrow \mathbf{c}^T \cdot \mathbf{g}$, $d \leftarrow \mathbf{d}^T \cdot \mathbf{g}$ and $\mathbf{f} \leftarrow \mathbf{F} \cdot \mathbf{g}$. Choose hash functions $\mathcal{H} : \mathbb{G} \to \{0,1\}^\lambda$ and $\mathcal{G} : \mathbb{G}^\ell \times \{0,1\}^\lambda \to \mathbb{Z}_q$ that will be modeled as random oracles. Return $\mathsf{pk} = (\mathcal{H}, \mathcal{G}, [\mathbf{g}, a, \mathbf{b}, c, d, \mathbf{f}])$, $\mathsf{sk} = (\mathbf{a}, \mathbf{B}, \mathbf{c}, \mathbf{d}, \mathbf{F})$.
- $\underline{\mathsf{Enc}(\mathsf{pk}, \bar{\mathtt{M}})}$: Sample $w, x, y \leftarrow\!\!\$\ \mathbb{Z}_q$ and $R \leftarrow\!\!\$\ \mathbb{G}$ uniformly at random, and compute:

$$\begin{array}{lll} \mathtt{ak} \leftarrow \mathcal{H}(R), & p \leftarrow \mathcal{G}(\bar{\mathtt{M}} \| \mathtt{ak}), & \\ [\mathbf{w}] \leftarrow w \cdot [\mathbf{g}], & [c_w] \leftarrow w \cdot [a] + R, & \bar{Z} \leftarrow w \cdot [\mathbf{f}] \\ [\mathbf{x}] \leftarrow x \cdot [\mathbf{g}], & [\mathbf{c}_x] \leftarrow x \cdot [\mathbf{b}] + \bar{\mathtt{M}}, & [p_x] \leftarrow x \cdot (p[c] + [d]), \\ [\mathbf{y}] \leftarrow y \cdot [\mathbf{g}], & [\mathbf{c}_y] \leftarrow y \cdot [\mathbf{b}], & [p_y] \leftarrow y \cdot (p[c] + [d]), \end{array}$$

  Define $\bar{W} := [\mathbf{w}, c_w] \in \mathbb{G}^3$, $\bar{X} := [\mathbf{x}, \mathbf{c}_x, p_x] + \bar{Z} \in \mathbb{G}^{\ell+3}$, $\bar{Y} := [\mathbf{y}, \mathbf{c}_y, p_y] \in \mathbb{G}^{\ell+3}$, and output the ciphertext $\mathtt{C} := (\bar{W}, \bar{X}, \bar{Y})$.
- $\underline{\mathsf{Rand}(\mathsf{pk}, \mathtt{C}; w', s, t)}$: Parse $\mathtt{C} = (\bar{W}, \bar{X}, \bar{Y})$ as defined above. Sample $w', t \leftarrow\!\!\$\ \mathbb{Z}_q$ and $s \leftarrow\!\!\$\ \mathbb{Z}_{2^\lambda}$ and output $\mathtt{C}' := (\bar{W}', \bar{X}', \bar{Y}')$ computed as follows:

$$\bar{W}' \leftarrow \bar{W} + w' \cdot [\mathbf{g}, a], \quad \bar{X}' \leftarrow \bar{X} + s \cdot \bar{Y} + w' \cdot [\mathbf{f}], \quad \bar{Y}' \leftarrow t \cdot \bar{Y}$$

- $\underline{\mathsf{Dec}(\mathsf{sk}, \mathtt{C})}$: Parse $\mathtt{C} := (\bar{W}, \bar{X}, \bar{Y})$, $\bar{W} := [\mathbf{w}, c_w]$.

Sample $t \leftarrow_{\$} \mathbb{Z}_q$, compute $\bar{X}' \leftarrow \bar{X} - \mathbf{F} \cdot [\mathbf{w}] + t \cdot \bar{Y}$, and parse $\bar{X}' := [\mathbf{x}, \mathbf{c}_x, p_x]$. Next, compute

$$\mathtt{ak} \leftarrow \mathcal{H}([c_w] - \mathbf{a}^T \cdot [\mathbf{w}]), \quad \bar{\mathtt{M}} \leftarrow [\mathbf{c}_x] - \mathbf{B} \cdot [\mathbf{x}], \quad p \leftarrow \mathcal{G}(\bar{\mathtt{M}} \| \mathtt{ak})$$

If $[c_w] - \mathbf{a}^T \cdot [\mathbf{w}] \neq [0]$ and $[p_x] = (p \cdot \mathbf{c} + \mathbf{d})^T \cdot [\mathbf{x}]$, output $\bar{\mathtt{M}}$, else output $\perp$.

The correctness of the scheme can be checked by inspection. If we do not consider re-randomization, then it simply reduces to the correctness property of the underlying SPHF, e.g., on that $\mathbf{a}^T \cdot [\mathbf{w}] = w \cdot [a]$ when $\mathbf{w} = w \cdot [\mathbf{g}]$.

We remark that the decryption procedure is randomized (namely, it sample the value t). In this way we can simultaneously check, by verifying only one equation, that both the $\bar{X}$ component and the $\bar{Y}$ component lie in the suitable subspaces. Alternatively, one could de-randomize the procedure by additionally check the validity of the p_y component of $\bar{Y}$ and that the $\mathbf{c}_y$ component decrypts to 0. Notice this extra checks on the $\bar{Y}$ component are necessary only to comply with our notion of correctness which asks that for any *valid* ciphertext a re-randomization of it decrypts to its intended value. However, one could redefine the decryption procedure by fixing $t = 0$ and still obtain a RCCA-secure scheme with a slightly relaxed notion of correctness.

In the following theorems we state the RCCA security (see Def. 9) and re-randomizability (see Def. 10) of the PKE scheme described above.

Theorem 2. *If the DDH assumption holds, the scheme $\mathcal{PKE}$ described above is RCCA secure in the NPRO Model.*

Theorem 3. *If the DDH assumption holds, the scheme $\mathcal{PKE}$ described above satisfies weak-RCCA re-randomizability in the NPRO Model.*

The proofs appear slightly later in Section 4.3.

Remark 1 (On Variations of the Scheme.). The proof of Theorem 2 does not use neither the component $\bar{Y}$ nor the secret key $\mathbf{F}$ to argue security. In fact, a PKE scheme like the above one but where ciphertexts consist only $(\bar{W}, \bar{X})$, and $y = 0$ in encryption is RCCA secure (but not re-randomizable). Golle *et al.* [29] define the notion of *universal re-randomizability* that says that the re-randomization algorithm can work without the public key. For efficiency reason, our PKE scheme is not universally re-randomizable (as, for example, the scheme of [43]). However, using the *double-strand* technique of Golle *et al.* for the component $\bar{W}$ (as we do with $\bar{X}, \bar{Y}$) we can make our scheme universally re-randomizable.

Our PKE with authenticated re-randomizability. Here we show that the scheme $\mathcal{PKE}$ described earlier can be extended to have authenticated re-randomizability. To this end, we describe the two additional algorithms $(\mathsf{ARand}, \mathsf{ADec})$, let $\mathcal{K} = \mathbb{Z}^*_{2^\lambda}$, and then show that the scheme $\mathcal{APKE} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{ARand}, \mathsf{ADec})$ has correctness, UF-CCA security, as well as RCCA security and re-randomizability.

- $\mathsf{ARand}(\mathsf{pk}, \kappa, \mathtt{C})$: Output $\mathtt{C}' \leftarrow \mathsf{Rand}(\mathsf{pk}, \kappa \cdot \mathtt{C})$.
- $\mathsf{ADec}(\mathsf{sk}, \kappa, \mathtt{C})$: Output $\mathsf{Dec}(\mathsf{sk}, \mathtt{C}/\kappa)$.

First, we show in the following lemma the correctness of the scheme.

Lemma 3. *The scheme $\mathcal{APKE}$ is a correct PKE scheme with authenticated re-randomizability. Namely, for any ℓ, any $\mathsf{sk}, \mathsf{pk} \in \mathsf{KGen}(\mathtt{prm}, 1^\ell)$ and for any message $\mathtt{M}$: (1) for any $\kappa \in \mathbb{Z}^*_q$ we have $\mathsf{ADec}(\mathsf{sk}, 1, \mathsf{Enc}(\mathsf{pk}, \mathtt{M})) = \mathtt{M}$; (2) for any $\kappa, \kappa' \in \mathbb{Z}^*_q$ and for any $\mathtt{C}$ such that $\mathsf{Dec}(\mathsf{sk}, \kappa, \mathtt{C}) = \mathtt{M}$ we have $\mathsf{ADec}(\mathsf{sk}, \kappa \cdot \kappa', \mathsf{ARand}(\mathsf{pk}, \kappa', \mathtt{C})) = \mathtt{M}$.*

Proof. Property (1) follows immediately by the correctness of $\mathcal{PKE}$. To see that the second correctness property holds, we proceed as follows.

Let $\tilde{\mathtt{C}} \leftarrow \mathsf{ARand}(\mathsf{pk}, \tilde{\kappa}, \mathtt{C})$, parse $\tilde{\mathtt{C}} = (\tilde{W}, \tilde{X}, \tilde{Y})$ with $\tilde{W} = [\tilde{\mathbf{w}}, \tilde{c}_w]$, $\tilde{Y} = [\tilde{\mathbf{y}}, \tilde{\mathbf{c}}_y, \tilde{p}_y]$, and parse analogously $\mathtt{C} = (\bar{W}, \bar{X}, \bar{Y})$. Let decryption of $\mathtt{C}$ with κ compute the following values:

$$\begin{aligned} R &\leftarrow ([c_w] - \mathbf{a}^T \cdot [\mathbf{w}])/\kappa \\ \bar{X}' &\leftarrow (\bar{X} - \mathbf{F} \cdot [\mathbf{w}])/\kappa \quad \text{and parse } \bar{X}' = [\mathbf{x}', \mathbf{c}'_x, p'_x] \\ \bar{\mathtt{M}} &\leftarrow [\mathbf{c}'_x] - \mathbf{B} \cdot [\mathbf{x}'] \end{aligned}$$

and let the check $[p_x] = (p \cdot \mathbf{c} + \mathbf{d})^T \cdot [\mathbf{x}]$ be satisfied, for $p = \mathcal{G}(\bar{\mathtt{M}} \| \mathcal{H}(R))$.

Decryption of the re-randomized ciphertext $\tilde{\mathtt{C}}$ with tag $\kappa \cdot \tilde{\kappa}$ instead computes the following:

$$\begin{aligned} \tilde{R} &\leftarrow ([\tilde{c}_w] - \mathbf{a}^T \cdot [\tilde{\mathbf{w}}])/(\kappa \cdot \tilde{\kappa}) \\ \tilde{X}' &\leftarrow (\tilde{X} - \mathbf{F} \cdot [\tilde{\mathbf{w}}])/(\kappa \cdot \kappa') \quad \text{and parse } \tilde{X}' = [\tilde{\mathbf{x}}', \tilde{\mathbf{c}}'_x, \tilde{p}'_x] \\ \tilde{\mathtt{M}} &\leftarrow [\tilde{\mathbf{c}}'_x] - \mathbf{B} \cdot [\tilde{\mathbf{x}}'] \end{aligned}$$

Let w, s, t be the randomizers used in ARand. First, we argue that $R' = R$:

$$\begin{aligned} \tilde{R} = ([\tilde{c}_w] - &\mathbf{a}^T \cdot [\tilde{\mathbf{w}}])/(\kappa \cdot \tilde{\kappa}) = \\ &= (\tilde{\kappa} \cdot [c_w] + w \cdot [a] - \mathbf{a}^T \cdot (\tilde{\kappa} \cdot [\mathbf{w}] + w \cdot [\mathbf{g}]))/(\kappa \cdot \tilde{\kappa}) \\ &= (\tilde{\kappa} \cdot [c_w] - \tilde{\kappa} \cdot \mathbf{a}^T \cdot [\mathbf{w}])/(\kappa \cdot \tilde{\kappa}) = ([c_w] - \mathbf{a}^T \cdot [\mathbf{w}])/\kappa = R \end{aligned}$$

Second, we argue that $\tilde{X}' = \bar{X}' + s \cdot \bar{Y}$:

$$\begin{aligned} \tilde{X}' &= (\tilde{X} - \mathbf{F} \cdot [\tilde{\mathbf{w}}])/(\kappa \cdot \tilde{\kappa}) = \\ &= (\tilde{\kappa}(\bar{X} + s \cdot \bar{Y} + s \cdot \bar{Y} + w \cdot [\mathbf{f}]) - \mathbf{F} \cdot \tilde{\kappa}([\mathbf{w}] + w \cdot [\mathbf{g}]))/(\kappa \cdot \tilde{\kappa}) = \\ &= (\bar{X} + s \cdot \bar{Y} - \mathbf{F} \cdot [\mathbf{y}])/\kappa = \bar{X} + s \cdot \bar{Y}. \end{aligned}$$

Third, we argue that $\tilde{\mathtt{M}} = \bar{\mathtt{M}}$:

$$[\tilde{\mathbf{c}}'_x] - \mathbf{B} \cdot [\tilde{\mathbf{x}}'] = [\tilde{\mathbf{c}}'_x] - \mathbf{B} \cdot [\tilde{\mathbf{x}}'] + s \cdot ([\mathbf{c}'_y] - \mathbf{B} \cdot [\mathbf{y}']) = [\mathbf{c}'_x] - \mathbf{B} \cdot [\mathbf{x}'] + s \cdot 0$$

Finally, by the same derivation in the equation above, the decryption checks $[\tilde{p}'_x] = (p \cdot \mathbf{c} + \mathbf{d})^T \cdot [\tilde{\mathbf{x}}']$ and $[\tilde{p}'_y] = (p \cdot \mathbf{c} + \mathbf{d})^T \cdot [\tilde{\mathbf{y}}']$ (with $p = \mathcal{G}(\bar{\mathtt{M}} \| \mathcal{H}(R))$) are also satisfied.

In the following theorems we state the security properties of the $\mathcal{APKE}$ scheme described above.

Theorem 4. *If the DDH assumption holds, the scheme $\mathcal{PKE}$ described above satisfies RCCA security and weak-RCCA re-randomizability in the NPRO Model.*

The proof of this theorem can be obtained in a straightforward way by adapting the proofs of theorems 2 and 3 for the scheme $\mathcal{PKE}$. The idea is very simple. Since the only difference is that for $\mathcal{APKE}$ the decryption oracle takes inputs $(\kappa, \mathtt{C})$, by following the construction of the ADec algorithm we observe that every oracle query to ADec can be simulated by using the Dec simulation on input $\mathtt{C}/\kappa$.

Theorem 5. *If the DDH assumption holds, the scheme $\mathcal{APKE}$ described above is UF-CCA-secure in the NPRO model.*

The proof of this theorem appears in Appendix 4.6. Here we give an intuition, which relies on two main ideas. The first one is that the re-randomization secret key κ is hidden to the adversary: via a series of hybrid steps based on the DDH assumption we can get to an experiment where each ciphertext is authenticated with a random and independent key κ_i. Once the previous step is done, the key κ used in the decryption of the ciphertext $\mathtt{C}^*$ is random and independent of the adversary's view. Hence, the second idea is that in such a case there is a negligible probability (over the choice of κ) that the adversary can create a valid ciphertext (unless it is a honest re-randomization of one it received).

4.4 Proof of Theorem 2 (RCCA Security)

Proof. We prove the theorem by defining the following sequence of hybrid experiments and arguing that each consecutive pair of experiments is indistinguishable. Eventually, in the last experiment the adversary can guess the challenge bit only with probability $1/2$.

To simplify the exposition, we consider a slightly different decryption algorithm where the randomizer t is always set to 0. Notice that for any adversary A for the RCCA experiment with oracle access to the original decryption procedure there exists an adversary A' for the RCCA experiment with oracle access to this new decryption procedure.

The reason is that the randomized part of the decryption algorithm can be publicly performed. More in details, the adversary A' emulates the adversary A and whenever it gets a decryption query $\mathtt{C} = (\bar{W}, \bar{X}, \bar{Y})$ it samples randomness $t \leftarrow\!\!\$\ \mathbb{Z}_q$ and sets $\mathtt{C}' = (\bar{W}, \bar{X} + t \cdot \bar{Y}, \bar{Y})$ and forwards $\mathtt{C}'$ to its decryption oracle. It is to see that the adversary A' simulates perfectly the original decryption oracle for the adversary A.

We are ready to start with the hybrids argument. In what follows we denote underlined the changes introduced in each experiment.

Hybrid $\mathbf{H}_1$. In this experiment $\mathbf{H}_1$ the challenge ciphertext $\mathtt{C}^* = (\bar{W}^*, \bar{X}^*, \bar{Y}^*)$ is encrypted as in $\mathbf{Exp}^{\mathtt{RCCA}}$ except that $[\mathbf{x}^*], [\mathbf{w}^*] \leftarrow\!\!\$\ \mathbb{G}^2 \setminus \mathit{Span}([\mathbf{g}])$ and the computation of $\mathtt{C}^*$ uses the private hashing procedure with knowledge of the secret material. Specifically:

$$\begin{aligned}
&\text{Sample: } \underline{[\mathbf{w}^*], [\mathbf{x}^*] \leftarrow\!\!\$\ \mathbb{G}^2 \setminus \mathit{Span}([\mathbf{g}])}, \quad y^* \leftarrow\!\!\$\ \mathbb{Z}_q \\
&\quad \text{let } \underline{[\mu_w] \leftarrow \mathbf{a}^T \cdot [\mathbf{w}^*]}, \underline{[\boldsymbol{\mu}_x] \leftarrow \mathbf{B} \cdot [\mathbf{x}^*]}, \\
&\\
&\quad \bar{W}^* \leftarrow ([\mathbf{w}^*], \underline{\mu_w} + R^*), \quad \bar{Z}^* \leftarrow \underline{\mathbf{F} \cdot [\mathbf{w}^*]} \\
&\quad \bar{X}^* \leftarrow ([\mathbf{x}^*], \underline{[\boldsymbol{\mu}_x]} + \bar{\mathtt{M}}_{b^*}, \underline{(p \cdot \mathbf{c} + \mathbf{d})^T \cdot [\mathbf{x}^*]}) + \bar{Z} \\
&\quad \bar{Y}^* \leftarrow y^* \cdot ([\mathbf{g}], \mathbf{b}, (p \cdot [c] + [d])
\end{aligned}$$

Lemma 4. *If the DDH Assumption holds over $\mathbb{G}$ as generated by* Setup *then* $\mathbf{Exp}^{\mathtt{RCCA}}_{\mathcal{PKE}} \approx_c \mathbf{H}_1$.

The proof is rather straightforward and thus we only give a sketch. The reduction, given in input a DDH challenge, which is either in $\mathit{Span}([\mathbf{g}])$ or random in $\mathbb{G}^2$, can sample all the secret material of the experiment (i.e., secret key values and challenge bit b^*) and runs an experiment that simulates either $\mathbf{Exp}^{\mathtt{RCCA}}_{\mathcal{PKE}}$ or $\mathbf{H}_1$ according to whether $[\mathbf{x}^*], [\mathbf{w}^*]$ are in $\mathit{Span}([\mathbf{g}])$ or random in $\mathbb{G}^2$. With a little more detail, this argument also relies on the fact that elements sampled from $\mathbb{G}^2$ and $\mathbb{G}^2 \setminus \mathit{Span}([\mathbf{g}])$ are statistically indistinguishable.

Hybrid $\mathbf{H}_2$. Let experiment $\mathbf{H}_2$ be the same as $\mathbf{H}_1$ but where all the randomness used for the challenge ciphertext, including $\mathtt{ak}^* \leftarrow\!\!\$\ \{0,1\}^\lambda$, is sampled at the beginning of $\mathbf{H}_2$, and where the decryption oracle executes the following decryption procedure:

$\underline{\mathsf{Dec}_2(\mathsf{sk}, \mathtt{C})}$: Parse $\mathtt{C} = (\bar{W}, \bar{X}, \bar{Y})$, and $\bar{W} = [\mathbf{w}, c_w]$. Proceed as Dec except that compute R as follow.
Let $\underline{\alpha_w, \beta_w \text{ be such that } \mathbf{w} = \alpha_w \cdot \mathbf{w}^* + \beta_w \cdot \mathbf{g}}$. Notice that every $\mathbf{w} \in \mathbb{Z}_q^2$ can be written in this way as $(\mathbf{w}^*, \mathbf{g})$ is a basis for $\mathbb{Z}_q^2$. $\underline{\text{Compute } R \leftarrow [c_w] - \alpha_w \cdot [\mu_w] - \beta_w \cdot [a]}$.

Lemma 5. *Hybrids* $\mathbf{H}_1$ *and* $\mathbf{H}_2$ *are identically distributed:* $\mathbf{H}_1 \equiv \mathbf{H}_2$.

Proof. The only difference between Dec and Dec_2 is that the former computes R as $[c_w] - \mathbf{a}^T \cdot [\mathbf{w}]$ while the latter computes as described above. However notice that:

$$\mathbf{a}^T \cdot [\mathbf{x}] = \mathbf{a}^T \cdot (\alpha_w [\mathbf{w}^*] + \beta [\mathbf{g}]) = \alpha_w (\mathbf{a}^T [\mathbf{w}^*]) + \beta_w \cdot [a] = \alpha_w \cdot [\mu_w] + \beta \cdot [a],$$

in the last equation, we use that $[\mu_w]$ is set as $\mathbf{a}^T \cdot [\mathbf{w}^*]$, as introduced in the experiment $\mathbf{H}_1$.

Hybrid $\mathbf{H}_3$. Let experiment $\mathbf{H}_3$ be the same as $\mathbf{H}_2$ but the element $\underline{\mu_w \leftarrow_\$ \mathbb{G}}$ is sampled uniformly at random.

Lemma 6. *The hybrids* $\mathbf{H}_3$ *and* $\mathbf{H}_3$ *are identically distributed:* $\mathbf{H}_3 \equiv \mathbf{H}_4$.

Proof. We use the non-adaptive smoothness of Corollary 1. In particular, notice that, given in input $(\mathtt{prm}, a, \mathbf{x}^*, \mu_w)$ we can perfectly simulate the experiment without the knowledge of the secret key componet $\mathbf{a}$. In fact, because of the changes introduced in $\mathbf{H}_3$, the decryption oracle does not use $\mathbf{a}$ to compute its answer.

Hybrid $\mathbf{H}_4$. Let experiment $\mathbf{H}_4$ be the same as $\mathbf{H}_3$ but where all the randomness used for the challenge ciphertext, including $\mathtt{ak}^* \leftarrow_\$ \{0,1\}^\lambda$, is sampled at the beginning of $\mathbf{H}_4$, and where the decryption oracle executes the following decryption procedure:

$\underline{\mathsf{Dec}_4(\mathsf{sk}, \mathtt{C})}$: Parse $\mathtt{C} = (\bar{W}, \bar{X}, \bar{Y})$, and $\bar{W} = [\mathbf{w}, c_w]$. Proceed as Dec except that, instead of computing R, it defines $\mathtt{ak}$ as follows.
Let α_w, β_w be such that $\mathbf{w} = \alpha_w \cdot \mathbf{w}^* + \beta_w \cdot \mathbf{g}$.
1. $\underline{\text{If } \alpha_w = 1 \text{ and } c_w = \alpha_w \cdot c_w^* + \beta_w \cdot a, \text{ then set } \mathtt{ak} \leftarrow \mathtt{ak}^*}$;
2. $\underline{\text{If } \alpha_w = 0 \text{ then compute } \mathtt{ak} \leftarrow \mathcal{H}([c_w] - \beta_w \cdot [a])}$;
3. $\underline{\text{If } \alpha_w \notin \{0,1\} \text{ or } (\alpha_w = 1 \text{ and } c_w \neq \alpha_w \cdot c_w^* + \beta_w \cdot a) \text{ then sample } \mathtt{ak} \leftarrow_\$ \{0,1\}^\lambda}$.

Lemma 7. *Hybrids* $\mathbf{H}_4$ *and* $\mathbf{H}_3$ *are statistically indistinguishable:* $\mathbf{H}_4 \approx_s \mathbf{H}_3$.

Proof. Let us call a ciphertext $\mathtt{C} = (\bar{W}, \bar{X}, \bar{Y})$ $\bar{W}$*-invalid* if it falls in case (3) of Dec_2, i.e., if $\alpha_w \notin \{0,1\}$ or $(\alpha_w = 1$ and $c_w \neq \alpha_w \cdot c_w^* + \beta_w \cdot a)$.

Let $\mathcal{Q}^{\mathcal{H}}$ be the set of queries made by A to the random oracle $\mathcal{H}$ together with the corresponding answers, and let $\hat{\mathcal{Q}}^{\mathcal{H}}$ be the projection of $\mathcal{Q}^{\mathcal{H}}$ to only the queries of $\mathcal{H}$. Let $\mathcal{Q}^{\mathsf{Dec}}$ be the queries made by A to the decryption oracle.

We define the following two events in the experiments $\mathbf{H}_3, \mathbf{H}_4$:

- $\mathtt{InvQuery}_{(a)}$: there exists a $\bar{W}$-invalid ciphertext $(\bar{W} = [\mathbf{w}, c_w], \bar{X}, \bar{Y}) \in \mathcal{Q}^{\mathsf{Dec}}$ such that $([c_w] - \mathbf{a}^T \cdot [\mathbf{w}]) \in \hat{\mathcal{Q}}^{\mathcal{H}}$
- $\mathtt{InvQuery}_{(b)}$: $R^* \in \hat{\mathcal{Q}}^{\mathcal{H}}$,

In the following claim we argue that $\mathbf{H}_4 \equiv \mathbf{H}_3$ unless either one of the above events occurs. Then we will conclude the proof by showing that each event occurs with negligible probability.

Claim 1 $|\Pr[\mathbf{H}_4] - \Pr[\mathbf{H}_3]| \leq \Pr\left[\mathtt{InvQuery}_{(a)}\right] + \Pr\left[\mathtt{InvQuery}_{(b)} \wedge \neg \mathtt{InvQuery}_{(a)}\right]$.

Proof. Let $\mathtt{E}$ be the event $(\mathtt{InvQuery}_{(a)} \vee (\mathtt{InvQuery}_{(b)} \wedge \neg\mathtt{InvQuery}_{(a)}))$. To prove the claim we rely on Shoup's difference lemma (see lemma 1) and a standard union bound. To apply this lemma, we have to show that $\Pr[\mathbf{H}_4 \wedge \neg\mathtt{E}] = \Pr[\mathbf{H}_3 \wedge \neg\mathtt{E}]$. Namely, since $\neg\mathtt{E} = \neg\mathtt{InvQuery}_{(a)} \wedge \neg\mathtt{InvQuery}_{(b)}$, we show that conditioned on the event that both $\mathtt{InvQuery}_{(a)}$ and $\mathtt{InvQuery}_{(b)}$ do not occur, the two hybrids are identically distributed.

Since the two games differ only in the answers to decryption queries, let us partition these queries in three classes: for $i = 1, 2, 3$, queries of type i are those that fall in the i-th case of the Dec_4 decryption algorithm. For queries of type 2, it is easy to see that they are answered in the same way in both games. For queries of type 1, the adversary has basically sent a component $\bar{W}$ that is a re-randomization of $[\mathbf{w}^*, c_w^*]$, which thus must decrypt to R^*. Therefore, in $\mathbf{H}_4$ continuing decryption with $\mathtt{ak} = \mathtt{ak}^*$ generates the same distribution as in $\mathbf{H}_3$ if we condition on the fact that $R^* \notin \hat{\mathcal{Q}}^{\mathcal{H}}$. For queries of type 3, Dec_4 answers by sampling $\mathtt{ak}$ at random that, similarly to the previous case, generates the same distribution as in $\mathbf{H}_3$ if we condition on the fact that $([c_w] - \mathbf{a}^T \cdot [\mathbf{w}]) \notin \hat{\mathcal{Q}}^{\mathcal{H}}$.

We continue the proof by bounding $\Pr\left[\mathtt{InvQuery}_{(a)}\right]$.

Claim 2 *For every PPT adversary* $\Pr\left[\mathtt{InvQuery}_{(a)}\right] \in \mathtt{negl}(\lambda)$.

Proof. We show that the claim holds over the random choice of $\mathbf{a} \in \mathbb{Z}_q^2$. Consider the following algorithm:

Adversary $\mathsf{B}(a)$:

1. Sample $\mathbf{B} \leftarrow_\$ \mathbb{Z}_q^{\ell\times 2}$, $\mathbf{c}, \mathbf{d} \leftarrow_\$ \mathbb{Z}_q^2$ and $\mathbf{F} \leftarrow_\$ \mathbb{Z}_q^{(\ell+3)\times 2}$ and set $\mathsf{sk} = (\perp, \mathbf{B}, \mathbf{c}, \mathbf{d}, \mathbf{F})$ and $\mathsf{pk} = [\mathbf{g}, a, \mathbf{b}, c, d, \mathbf{f}]$ where $[\mathbf{b}] = [\mathbf{B} \cdot \mathbf{g}]$, $[c] = [\mathbf{c}^T \cdot \mathbf{g}]$, $[d] = [\mathbf{d}^T \cdot \mathbf{g}]$ and $[\mathbf{f}] = [\mathbf{F} \cdot \mathbf{g}]$.
2. Run the hybrid experiment $\mathbf{H}_4$ with the adversary A where the decryption oracle Dec_4 takes as secret key sk, and where the random oracle $\mathcal{H}$ is simulated in the non-programmable way (recall, this means that the reduction can only see the queries made by A but it cannot program their outputs). Let $\mathcal{Q}^{\mathsf{Dec}}$ be the set of all queries made by A to the decryption oracle. Notice that the challenge ciphertext can be sampled without the knowledge of $\mathbf{a}^T \cdot [\mathbf{w}^*]$ because of the change introduced in $\mathbf{H}_3$.
3. Pick a random $\mathtt{C}$ from $\mathcal{Q}^{\mathsf{Dec}}$ and pick a random element $(R, \mathtt{ak})$ from $\mathcal{Q}^{\mathcal{H}}$. Parse $\mathtt{C} = (\bar{W}, \bar{X}, \bar{Y})$ where $\bar{W} = [\mathbf{w}, c_w]$ and output $([\mathbf{w}], [c_w] - R)$.

We show that

$$\Pr_{\mathbf{a}\leftarrow_\$ \mathbb{Z}_q^2, \mathsf{B}}\left[\begin{array}{c|c} \mathbf{a}^T \cdot [\mathbf{w}] = [c_w] - R & \mathbf{a}^T \cdot \mathbf{g} = a \\ \mathbf{w} \notin Span(\mathbf{g}) & \end{array}\right] \geq \frac{\Pr\left[\mathtt{InvQuery}_{(a)}\right]}{|\mathcal{Q}^{\mathsf{Dec}}| \cdot |\mathcal{Q}^{\mathcal{H}}|}$$

In fact, conditioning on $\mathtt{InvQuery}_{(a)}$, with probability $1/|\mathcal{Q}^{\mathsf{Dec}}|$ the ciphertext $\mathtt{C} = (\bar{W}, \bar{X}, \bar{Y})$ chosen by B is such that $([c_w] - \mathbf{a}^T[\mathbf{w}]) \in \mathcal{Q}^{\mathcal{H}}$. Conditioning on the latter, with probability $1/|\mathcal{Q}^{\mathcal{H}}|$ the element $(R, \mathtt{ak})$ is such that $R = [c_w] - \mathbf{a}^T \cdot [\mathbf{w}]$.

Therefore, by applying Lemma 2, we obtain $\Pr\left[\mathtt{InvQuery}_{(a)}\right] \leq |\mathcal{Q}^{\mathsf{Dec}}| \cdot |\mathcal{Q}^{\mathcal{H}}|/q$, and since both $|\mathcal{Q}^{\mathcal{H}}|$ and $|\mathcal{Q}^{\mathsf{Dec}}|$ are polynomially bounded in λ, we obtain our claim.

We show that $\Pr\left[\mathtt{InvQuery}_{(b)} \wedge \neg\mathtt{InvQuery}_{(a)}\right]$ is negligible in λ over the randomness of $\mathbf{a}$.

Claim 3 *For every PPT adversary* $\Pr\left[\mathtt{InvQuery}_{(b)} \wedge \neg\mathtt{InvQuery}_{(a)}\right] \in \mathtt{negl}(\lambda)$.

Proof. This proof is similar to the one of the previous claim. Consider the following algorithm:

Adversary $\mathsf{B}(a)$:

1. Sample $\mathbf{B} \leftarrow_\$ \mathbb{Z}_q^{\ell\times 2}$, $\mathbf{c}, \mathbf{d} \leftarrow_\$ \mathbb{Z}_q^2$ and $\mathbf{F} \leftarrow_\$ \mathbb{Z}_q^{(\ell+3)\times 2}$ and set $\mathsf{sk} = (\perp, \mathbf{B}, \mathbf{c}, \mathbf{d}, \mathbf{F})$ and $\mathsf{pk} = [\mathbf{g}, a, \mathbf{b}, c, d, \mathbf{f}]$ where $[\mathbf{b}] = [\mathbf{B}\cdot\mathbf{g}]$, $[c] = [\mathbf{c}^T\cdot\mathbf{g}]$, $[d] = [\mathbf{d}^T\cdot\mathbf{g}]$ and $[\mathbf{f}] = [\mathbf{F}\cdot\mathbf{g}]$.
2. Run the hybrid experiment $\mathbf{H}_4$ with the adversary A where the decryption oracle Dec_4 takes as secret key sk, the random oracle $\mathcal{H}$ is simulated in the non-programmable way, and with the only difference that the challenge ciphertext component $[c_w^*]$ is sampled uniformly at random from $\mathbb{G}$.
3. Pick a random element $(R, \mathtt{ak})$ from $\mathcal{Q}^{\mathcal{H}}$ and output $([\mathbf{w}], [c_w^*] - R)$.

We show that

$$\Pr_{\mathbf{a}\leftarrow_\$ \mathbb{Z}_q^2, \mathsf{B}}\left[\begin{array}{c}\mathbf{a}^T\cdot[\mathbf{w}] = [c_w^*] - R \\ \mathbf{w}\notin Span(\mathbf{g})\end{array}\middle|\ \mathbf{a}^T\cdot\mathbf{g} = a\right] \geq \frac{\Pr\left[\mathtt{InvQuery}_{(b)} \wedge \neg\mathtt{InvQuery}_{(a)}\right]}{|\mathcal{Q}^{\mathcal{H}}|}$$

In fact, first notice, conditioning on $\neg\mathtt{InvQuery}_{(a)}$ then by Corollary 1 the distribution of $[c_w^*]$ in $\mathbf{H}_4$ is equivalent to the uniform distribution over $\mathbb{G}$. Further conditioning on $\mathtt{InvQuery}_{(b)}$, with probability $1/|\mathcal{Q}^{\mathcal{H}}|$ the element $(R, \mathtt{ak})$ is such that $R = ([c_w^*] - \mathbf{a}^T\cdot[\mathbf{w}^*]) = R^*$.

By Lemma 2 the left hand side of the above equation is upper bounded by $1/q$. Therefore we obtain that $\Pr\left[\mathtt{InvQuery}_{(b)} \wedge \neg\mathtt{InvQuery}_{(a)}\right] \leq |\mathcal{Q}^{\mathcal{H}}|/q$, and since $|\mathcal{Q}^{\mathcal{H}}|$ is polynomially bounded in λ, we obtain our claim.

By combining the three Claims above we obtain the proof of Lemma 7.

Hybrid $\mathbf{H}_5$. Let experiment $\mathbf{H}_5$ be the same as $\mathbf{H}_4$ but with the decryption algorithm modified as follow:

$\mathsf{Dec}_5(\mathsf{sk}, \mathtt{C})$: Parse $\mathtt{C} = (\bar{W}, \bar{X}, \bar{Y})$ and $\bar{Y} = [\mathbf{y}, \mathbf{c}_y, p_y]$. Compute $\mathtt{ak}$ from $\bar{W}$ as in Dec_4, and let $[\mathbf{x}, \mathbf{c}_x, p_x] \leftarrow \bar{X} - \bar{Z}$.
Let $\alpha, \beta \in \mathbb{Z}_q$ be such that $\mathbf{x} = \alpha\cdot\mathbf{x}^* + \beta\cdot\mathbf{g}$. Notice that, as in $\mathbf{H}_2$, every $\mathbf{x}$ can be written in this way.
Compute $\bar{\mathtt{M}} \leftarrow [\mathbf{c}_x] - (\alpha\cdot\boldsymbol{\mu}_x + \beta\cdot[\mathbf{b}])$;
If $[p_x] = (p\cdot\mathbf{c} + \mathbf{d})^T\cdot[\mathbf{x}]$, output $\bar{\mathtt{M}}$, else output $\perp$.

Lemma 8. *The hybrids $\mathbf{H}_5$ and $\mathbf{H}_4$ are identically distributed: $\mathbf{H}_5 \equiv \mathbf{H}_4$.*

Proof. The only difference between Dec_4 and Dec_5 is that the former computes $\bar{\mathtt{M}}$ as $[\mathbf{c}_x] - \mathbf{B}\cdot[\mathbf{x}]$ while the latter computes it as described above. However notice that:

$$\mathbf{B}\cdot[\mathbf{x}] = \mathbf{B}\cdot(\alpha\cdot[\mathbf{x}^*] + \beta\cdot[\mathbf{g}]) = \alpha\cdot(\mathbf{B}\cdot[\mathbf{x}^*]) + \beta\cdot[\mathbf{b}] = \alpha\cdot\boldsymbol{\mu}_x + \beta\cdot[\mathbf{b}],$$

in the last equation, we use that $\boldsymbol{\mu}_x$ is set $\mathbf{B}\cdot[\mathbf{x}^*]$, as introduced in the experiment $\mathbf{H}_1$.

Hybrid $\mathbf{H}_6$. Let experiment $\mathbf{H}_6$ be the same as $\mathbf{H}_5$ but the element $\boldsymbol{\mu}_x \leftarrow_\$ \mathbb{G}^\ell$ is sampled uniformly at random.

Lemma 9. *The hybrids $\mathbf{H}_6$ and $\mathbf{H}_5$ are identically distributed: $\mathbf{H}_6 \equiv \mathbf{H}_5$.*

Proof. We use the non-adaptive smoothness of Corollary 1. In particular, notice that, given in input $(\mathtt{prm}, \mathbf{b}, \mathbf{x}^*, \boldsymbol{\mu}_x)$ we can perfectly simulate the experiment without the knowledge of the secret key componet $\mathbf{B}$. In fact, because of the changes introduced in $\mathbf{H}_5$, the decryption oracle does not use $\mathbf{B}$ to compute its answer.

Hybrid $\mathbf{H}_7$. Let experiment $\mathbf{H}_7$ be the same as $\mathbf{H}_6$ but with the decryption algorithm modified as follow:

$\underline{\mathsf{Dec}_7(\mathsf{sk}, \mathtt{C})}$: Parse $\mathtt{C} = (\bar{W}, \bar{X}, \bar{Y})$. Compute $\mathtt{ak}$ from $\bar{W}$ as in Dec_4. Let $\alpha, \beta \in \mathbb{Z}_q$ be such that $\mathbf{x} = \alpha \cdot \mathbf{x}^* + \beta \cdot \mathbf{g}$, and compute $\bar{\mathtt{M}}$ as in the previous experiment.
$\underline{\text{If } \mathtt{ak} \neq \mathtt{ak}^* \text{ and } \alpha \neq 0, \text{ or } \mathtt{ak} = \mathtt{ak}^* \text{ and } \alpha \neq 0 \text{ and } \bar{\mathtt{M}} \notin \{\bar{\mathtt{M}}_0, \bar{\mathtt{M}}_1\} \text{ then output } \perp,}$
$\underline{\text{else, execute the last line of } \mathsf{Dec}_5.}$

Lemma 10. *The hybrids $\mathbf{H}_7$ and $\mathbf{H}_6$ are statistically indistinguishable: $\mathbf{H}_7 \approx_s \mathbf{H}_6$.*

Proof. Let `BadDecryption` be the event that the adversary queries the decryption oracle on a ciphertext such that $\mathtt{ak} \neq \mathtt{ak}^*$ and $\alpha \neq 0$, or $\mathtt{ak} = \mathtt{ak}^*$ and $\alpha \neq 0$ and $\bar{\mathtt{M}} \notin \{\bar{\mathtt{M}}_0, \bar{\mathtt{M}}_1\}$, and that in experiment $\mathbf{H}_6$ would be answered with an output $\neq \perp$.

Claim 4 $|\Pr[\mathbf{H}_7] - \Pr[\mathbf{H}_6]| \leq \Pr[\mathtt{BadDecryption}]$.

Proof. We rely again on Shoup's difference lemma (see lemma 1). The two games differ only in the answers to decryption queries, and in particular, while Dec_7, for the queries described by the event above outputs $\perp$, Dec_5 may not.

We show that `BadDecryption` happens with probability negligible in the security parameter.

First, notice that if either $\mathtt{ak} \neq \mathtt{ak}^*$ and $\alpha \neq 0$, or $\mathtt{ak} = \mathtt{ak}^*$ and $\alpha \neq 0$ and $\bar{\mathtt{M}} \notin \{\bar{\mathtt{M}}_0, \bar{\mathtt{M}}_1\}$ happen, then the value $p \leftarrow \mathcal{G}(\bar{\mathtt{M}} \| \mathtt{ak})$ is different from $p^* = \mathcal{G}(\bar{\mathtt{M}}_{b^*} \| \mathtt{ak}^*)$ with overwhelming probability. This is easy to see: in the first case $\mathtt{ak} \neq \mathtt{ak}^*$ and therefore $\mathtt{ak} \neq \mathtt{ak}^*$; in the second case $\bar{\mathtt{M}} \notin \{\bar{\mathtt{M}}_0, \bar{\mathtt{M}}_1\}$.

By definition of `BadDecryption`, the decryption oracle in $\mathbf{H}_6$ does not output $\perp$ therefore:

$$\begin{aligned}[p_x] =& (p \cdot \mathbf{c} + \mathbf{d})^T \cdot [\mathbf{x}] \\ =& (p \cdot \mathbf{c} + \mathbf{d})^T \cdot [\alpha \cdot \mathbf{x}^* + \beta \cdot \mathbf{g}] \\ =& (p \cdot \mathbf{c} + \mathbf{d})^T \cdot \alpha \cdot [\mathbf{x}^*] + \beta \cdot (p \cdot [c] + [d])\end{aligned}$$

If we let $[\gamma] \leftarrow \mathbf{c}^T[\mathbf{x}^*]$ and $[\delta] \leftarrow \mathbf{d}^T[\mathbf{x}^*]$, we can rewrite the above equation as:

$$p \cdot [\gamma] + [\delta] = ([p_x] - \beta \cdot (p \cdot [c] + [d]))/\alpha \tag{1}$$

Below we argue that this equation is satisfied with negligible probability.

By Corollary 1, one can see that, before the adversary makes any query, $[\gamma, \delta]$ are uniformly distributed in $\mathbb{G}^2$. After seeing the challenge ciphertext and interacting with the decryption oracle, the adversary can learn some information about γ and δ. Yet, we argue that conditioned on this information, γ and δ are sufficiently random to make the event happen with negligible probability.

More precisely, let j be the index of the decryption query such that the event `BadDecryption` happens for the first time. This means that for the i-th query with $i < j$, and where the ciphertext is of the form described in `BadDecryption`, both $\mathbf{H}_7$ and $\mathbf{H}_6$ would output $\perp$. Notice that at the time of the j-th decryption query, the adversary has learned some information about γ and δ. Let us consider every previous decryption query $i < j$. We have three cases:

1. If the queried ciphertext has $\alpha_i = 0$, the query's outcome reveals no information on γ, δ.
2. If the queried ciphertext has $\alpha_i \neq 0$ and $\mathtt{ak}_i = \mathtt{ak}^*$ and $\bar{\mathtt{M}}_i = \bar{\mathtt{M}}_{b^*}$, then the value p_i computed by the decryption oracle is equal to p^* which is already present in the view of the adversary. The latter implies that no information on γ, δ is revealed.

3. If the queried ciphertext is such that $\mathtt{ak}_i \neq \mathtt{ak}^*$ and $\alpha_i \neq 0$, or $\mathtt{ak}_i = \mathtt{ak}^*$ and $\alpha_i \neq 0$ and $\bar{\mathtt{M}}_i \neq \bar{\mathtt{M}}_{b^*}$, then the adversary learns that $p_i \cdot \gamma + \delta \neq p_{x,i} - \beta_i \cdot (p_i \cdot c + d)/\alpha_i$, where $p_{x,i}, p_i, \beta_i, \alpha_i$ are the values p_x, p, β, α as computed in the i-th decryption.

The above considerations and the definition of γ and δ implies that the distribution of $[\gamma, \delta]$ at the time of the j-th query is uniform over the set

$$\mathcal{S}_j = \{[\gamma, \delta] \in \mathbb{G}^2 : p^* \cdot \gamma + \delta = p_x^* \bigwedge_{i<j} p_i \cdot \gamma + \delta \neq p_{x,i} - \beta_i \cdot (p_i \cdot c + d)/\alpha_i \wedge p^* \neq p_i\}$$

which has cardinality $\geq q - j$.

Therefore, considering the probabilities that $\mathtt{ak} \neq \mathtt{ak}^*$ and that equation (1) is satisfied, we have that `BadDecryption` occurs for the first time in the j-th decryption query with probability at most $(1 - 2^{-\lambda})/(q - j)$. By a union bound, we can conclude that $\Pr[\mathtt{BadDecryption}] \leq |\mathcal{Q}^{\mathsf{Dec}}|(1-2^{-\lambda})/(q-|\mathcal{Q}^{\mathsf{Dec}}|)$. Finally, notice that $|\mathcal{Q}^{\mathsf{Dec}}|$ is a polynomial in λ while q is exponential in λ. Hence this probability is negligible in λ.

Hybrid $\mathbf{H}_8$. Let experiment $\mathbf{H}_8$ be the same as $\mathbf{H}_7$ but with the decryption algorithm modified as follow:

> $\underline{\mathsf{Dec}_8(\mathsf{sk}, \mathtt{C}):}$ Parse $\mathtt{C} = (\bar{W}, \bar{X}, \bar{Y})$. Compute $\mathtt{ak}$ from $\bar{W}$ as in Dec_4. Let $\alpha, \beta \in \mathbb{Z}_q$ be such that $\mathbf{x} = \alpha \cdot \mathbf{x}^* + \beta \cdot \mathbf{g}$, and compute $\bar{\mathtt{M}}$ as in the previous experiment. Compute $p \leftarrow \mathcal{G}(\bar{\mathtt{M}} \| \mathtt{ak})$.
> If $\mathtt{ak} \neq \mathtt{ak}^*$ and $\alpha \neq 0$, or $\mathtt{ak} = \mathtt{ak}^*$ and $\alpha \neq 0$ and $\bar{\mathtt{M}} \notin \{\bar{\mathtt{M}}_0, \bar{\mathtt{M}}_1\}$ then output $\perp$ (as in the previous experiment),
> $\underline{\text{else if (1) } \mathtt{ak} = \mathtt{ak}^* \text{ and } \alpha \notin \{0, 1\} \text{ and } \bar{\mathtt{M}} \in \{\bar{\mathtt{M}}_0, \bar{\mathtt{M}}_1\}, \text{ or (2) } \mathtt{ak} = \mathtt{ak}^* \text{ and } \alpha = 0, \text{ then output } \perp,}$
> else, execute the last line of Dec_5.

Lemma 11. *The hybrids $\mathbf{H}_8$ and $\mathbf{H}_7$ are statistically indistinguishable: $\mathbf{H}_8 \approx_s \mathbf{H}_7$.*

Proof. Let $\mathtt{BadDecryption}(j)$ be the event that the j-th query to the decryption oracle is the first one such that either $\mathtt{ak} = \mathtt{ak}^*$ and $\alpha \notin \{0, 1\}$ and $\bar{\mathtt{M}} \in \{\bar{\mathtt{M}}_0, \bar{\mathtt{M}}_1\}$ or $\mathtt{ak} = \mathtt{ak}^*$ and $\alpha = 0$, and that in experiment $\mathbf{H}_7$ the query would be answered with an output $\neq \perp$.

Claim 5 $|\Pr[\mathbf{H}_8] - \Pr[\mathbf{H}_7]| \leq \Pr\left[\exists j \leq |\mathcal{Q}^{\mathsf{Dec}}| : \mathtt{BadDecryption}(j)\right]$.

The claims follows easily applying the Shoup's difference lemma (see lemma 1). Recalling that $\mathcal{Q}^{\mathcal{G}}$ is the set of queries to the random oracle $\mathcal{G}$, define the event `Queried` be the event that $\exists \bar{\mathtt{M}}$ such that $(\bar{\mathtt{M}}, \mathtt{ak}) \in \mathcal{Q}^{\mathcal{G}}$ notice that:

$$\Pr[\mathtt{BadDecryption}(j)] \leq \Pr[\mathtt{BadDecryption}(j) | \neg\mathtt{Queried}] + \Pr[\mathtt{Queried}].$$

Claim 6 $\Pr[\mathtt{BadDecryption}(j) | \neg\mathtt{Queried}] \leq \mathtt{negl}(\lambda)$.

Proof. If the event $\mathtt{BadDecryption}(j)$ happens then it holds $p_x = (p \cdot \mathbf{c} + \mathbf{d})^T \cdot \mathbf{x}$ where $p \leftarrow \mathcal{G}(\bar{\mathtt{M}} \| \mathtt{ak}^*)$. However, since $\neg\mathtt{Queried}$ then the value p is uniformly distributed over $\mathbb{Z}_q$. Hence, for such random p (and considering that $\mathbf{x} \neq \mathbf{0}$ since $\alpha \neq 0$) the equation $p_x = (p \cdot \mathbf{c} + \mathbf{d})^T \cdot \mathbf{x}$ holds with probability $\leq 1/q$.

Let $\mathcal{Q}^{\mathsf{Dec}}$ be the set of decryption oracle queries.

Claim 7 $\Pr[\mathtt{Queried}] \leq |\mathcal{Q}^{\mathcal{H}'}|/(2^\lambda - |\mathcal{Q}^{\mathsf{Dec}}|)$.

Proof. Notice that $\mathtt{ak}^*$ is sampled uniformly from $\{0,1\}^\lambda$, because of the change introduced in $\mathbf{H}_4$, moreover, $\mathtt{ak}^*$ is independent from $\mathtt{C}^*$, because of the change introduced in experiment $\mathbf{H}_6$. However, we cannot claim that $\mathtt{ak}^*$ is uniformly distributed over $\{0,1\}^\lambda$ given all the view of the adversary. In fact, the behavior of Dec_8 depends on $\mathtt{ak}^*$. However, below we argue that the decryption oracle leaks only a small amount of information about $\mathtt{ak}^*$. We analyze this case by case:

1. If $\alpha = 0$ then when $\mathtt{ak} = \mathtt{ak}^*$ the decryption algorithm always output $\perp$, on the other hand, when $\mathtt{ak} \neq \mathtt{ak}^*$, the decryption could output either $\perp$ or a message $\bar{\mathtt{M}}'$. Notice that the message is computed as a function of $\mathtt{ak}$ which is uniformly random conditioned on $\mathtt{ak} \neq \mathtt{ak}^*$, therefore the information that $\bar{\mathtt{M}}'$ carries about $\mathtt{ak}^*$ is not more than the information of $\mathtt{ak}$ carries about $\mathtt{ak}^*$, which is just that $\mathtt{ak} \neq \mathtt{ak}^*$.
2. If $\alpha = 1$ then when $\mathtt{ak} = \mathtt{ak}^*$ the decryption oracle outputs either $\diamond$ or $\perp$, on the other hand, when $\mathtt{ak} \neq \mathtt{ak}^*$, the decryption outputs $\perp$.
3. If $\alpha \notin \{0,1\}$ then when $\mathtt{ak} = \mathtt{ak}^*$ then the decryption oracle outputs $\perp$, also, when $\mathtt{ak} \neq \mathtt{ak}^*$, the decryption outputs $\perp$.

By the analysis above, all the queries made by the adversary allow to exclude one possible assignment for the value of $\mathtt{ak}^*$. Specifically, for (1) the worst case is when in one case the decryption oracle outputs $\perp$ but in the other case outputs a message $\bar{\mathtt{M}}'$, for (2) the worst case is when in one case the decryption oracle outputs $\diamond$ but in in the other case outputs $\perp$ and for (3) the decryption oracle gives no information since it always outputs $\perp$. The random variable $\mathtt{ak}^*$ is uniformly distributed over a space of size $2^\lambda - |\mathcal{Q}^{\mathsf{Dec}}|$, therefore for any fixed $\mathtt{ak}'$ the probability that $\mathtt{ak}^*$ is equal to $\mathtt{ak}'$ given the view is $1/(2^\lambda - |\mathcal{Q}^{\mathsf{Dec}}|)$. By a simple union bound over all the random oracle query to $\mathcal{H}'$, we can prove the statement of the claim.

Putting together Claim 6 and Claim 7, and by union bound over all the queries to the decryption oracle we obtain that the probability that exists $j \leq |\mathcal{Q}^{\mathsf{Dec}}|$ such that $\mathtt{BadDecryption}(j)$ is negligible in λ and therefore, by Claim 5, the statement of the lemma.

Hybrid $\mathbf{H}_9$. Let experiment $\mathbf{H}_9$ be the same as $\mathbf{H}_8$ but with the decryption algorithm modified as follow:

> $\mathsf{Dec}_8(\mathsf{sk}, \mathtt{C})$: Parse $\mathtt{C} = (\bar{W}, \bar{X}, \bar{Y})$. Compute $\mathtt{ak}$ from $\bar{W}$ as in Dec_4. Let $\alpha, \beta \in \mathbb{Z}_q$ be such that $\mathbf{x} = \alpha \cdot \mathbf{x}^* + \beta \cdot \mathbf{g}$, and compute $\bar{\mathtt{M}}$ as in the previous experiment. Compute $p \leftarrow \mathcal{G}(\bar{\mathtt{M}} \| \mathtt{ak})$.
> If $\mathtt{ak} \neq \mathtt{ak}^*$ and $\alpha \neq 0$, or $\mathtt{ak} = \mathtt{ak}^*$ and $\alpha \neq 0$ and $\bar{\mathtt{M}} \notin \{\bar{\mathtt{M}}_0, \bar{\mathtt{M}}_1\}$ then output $\perp$ (as in the previous experiment),
> else if (1) $\mathtt{ak} = \mathtt{ak}^*$ and $\alpha \notin \{0,1\}$ and $\bar{\mathtt{M}} \in \{\bar{\mathtt{M}}_0, \bar{\mathtt{M}}_1\}$, or (2) $\mathtt{ak} = \mathtt{ak}^*$ and $\alpha = 0$, then output $\perp$,
> else if $\mathtt{ak} = \mathtt{ak}^*$ and $\alpha = 1$ and $\bar{\mathtt{M}} \in \{\bar{\mathtt{M}}_0, \bar{\mathtt{M}}_1\}$ then
> let $[\tilde{\mathbf{x}}, \tilde{\mathbf{c}}_x, \tilde{p}_x] \leftarrow \bar{X}^* - \bar{Z}^*$, (where $\bar{X}^*$, $\bar{Z}^*$ are defined in $\mathbf{H}_1$)
> if $[\mathbf{c}_x, p_x] = [\tilde{\mathbf{c}}_x, \tilde{p}_x] + \beta \cdot [\mathbf{b}, p^*c + d]$ return $\diamond$ else $\perp$,
> else execute the last line of Dec_5.

Lemma 12. *The hybrids $\mathbf{H}_9$ and $\mathbf{H}_8$ are statistically indistinguishable: $\mathbf{H}_9 \approx_s \mathbf{H}_8$.*

Proof. Let $\mathtt{BadDecryptionSame}$ be the event that the adversary queries the decryption oracle with a ciphertext such that $\mathtt{ak} = \mathtt{ak}^*$ and $\alpha = 1$ and $\bar{\mathtt{M}} \in \{\bar{\mathtt{M}}_0, \bar{\mathtt{M}}_1\}$, and $[\mathbf{c}_x, p_x] \neq [\tilde{\mathbf{c}}_x, \tilde{p}_x] + \beta \cdot [\mathbf{b}, p^*c + d]$ but the decryption oracle Dec_8 would not output $\perp$.

Claim 8 $|\Pr[\mathbf{H}_9] - \Pr[\mathbf{H}_8]| \leq \Pr[\texttt{BadDecryptionSame}]$.

Proof. We rely again on Shoup's difference lemma. The two games might differ only when answering decryption queries such that $\texttt{ak} = \texttt{ak}^*$ and $\alpha = 1$ and $\bar{\texttt{M}} \in \{\bar{\texttt{M}}_0, \bar{\texttt{M}}_1\}$. Moreover, notice that if $[\mathbf{c}_x, p_x] = [\tilde{\mathbf{c}}_x, \tilde{p}_x] + \beta \cdot [\mathbf{b}, p^*c + d]$ then both decryption oracles would answer $\diamond$, so the relative branch in the decryption procedure Dec_9 returns the same as Dec_8 would. So the two experiments proceed exactly the same conditioned on the event $\texttt{BadDecryptionSame}$ not happening.

We show that $\texttt{BadDecryptionSame}$ happens with negligible probability in the security parameter. Notice that since $[\mathbf{c}_x, p_x] \neq [\tilde{\mathbf{c}}_x, \tilde{p}_x] + \beta \cdot [\mathbf{b}, p^*c + d]$ but $\texttt{ak} = \texttt{ak}^*$, and $\alpha = 1$ and $\bar{\texttt{M}} \in \{\bar{\texttt{M}}_0, \bar{\texttt{M}}_1\}$, and the ciphertext decrypt correctly in $\mathbf{H}_8$, then it must be that $\bar{\texttt{M}} = \bar{\texttt{M}}_{1-b^*}$, in fact if $[\mathbf{c}_x] = [\tilde{\mathbf{c}}_x]$ then $p_x \neq \tilde{p}_x + p^*c + d$ and thus the ciphertext cannot decrypt correctly.

As shown in the previous lemma, $\Pr\left[\exists \bar{\texttt{M}} : (\bar{\texttt{M}}, \texttt{ak}^*) \in \mathcal{Q}^{\mathcal{G}}\right] \in \texttt{negl}(\lambda)$ in $\mathbf{H}_8$. So, let $p' \leftarrow \mathcal{G}(\bar{\texttt{M}}_{1-b^*} \| \texttt{ak}^*)$ then p' is statistically close to a value uniformly distributed over $\mathbb{Z}_q$. Given this, the equation $p_x = (p' \cdot \mathbf{c} + \mathbf{d})^T \cdot \mathbf{x}$ holds with negligible probability, which implies that the probability of $\texttt{BadDecryptionSame}$ is negligible.

Lemma 13. *In* $\mathbf{H}_9$, $\Pr[b_{\mathsf{A}} = b^*] = \frac{1}{2}$

Proof. We show that in $\mathbf{H}_9$ the $\Pr[b_{\mathsf{A}} = b^*]$ is equal to $\frac{1}{2}$. In fact, since $\boldsymbol{\mu}_x$ is chosen uniformly at random, the ciphertext $\texttt{C}^*$ and the bit b^* are independently distributed given the public key and all the answers to the oracle queries up to generation of the $\texttt{C}^*$. Moreover, all the queries with $\texttt{ak} \neq \texttt{ak}^*$ and $\alpha \neq 0$ are answered with $\perp$, which does not give any further information about b^*; all the queries with $\texttt{ak} \neq \texttt{ak}^*$ and $\alpha = 0$ can be answered as a function of the view of the adversary. Finally, all the queries with $\texttt{ak} = \texttt{ak}^*$ are answered either with $\perp$ or with $\diamond$. Notice that $\diamond$ is given independently of b^*. Wrapping up all together, the full view of the adversary in the experiment $\mathbf{H}_9$ is independent of the challenge bit, therefore the lemma follows.

By the lemmas above and the triangular inequality the distribution of the real experiment is $\frac{1}{2} + \texttt{negl}(\lambda)$. Moreover, all the reductions given do not need to program the random oracle but instead they simply keep track of the queries made by the adversary. Therefore the scheme is secure in the Non-Programmable Random Oracle Model.

4.5 Proof of Theorem 3 (re-randomizability)

Proof. We prove the theorem using a sequence of hybrid experiments.

Similarly to the proof of Theorem 2, to simplify the exposition, we consider a slightly different decryption algorithm where the randomizer t is always set to 0. As before, for any adversary A for the Rand-wRCCA experiment for the original PKE scheme there exists an adversary A' for the Rand-wRCCA experiment over this slightly different PKE scheme that has the same advantage of A. The reason is the same as before, namely, that the randomized part of the decryption algorithm can be publicly performed. There is a little detail to notice. In the Rand-wRCCA experiment the adversary sends the challenge ciphertext $\texttt{C}$ (see Fig. 4), the adversary looses immediately if the ciphertext does not decrypt. However, we notice that if the challenge ciphertext $\texttt{C}$ decrypts under the randomized decryption procedure then it does decrypt under the slightly modified decryption procedure. We can do this without loss of generality because the randomized part of the decryption procedure can be publicly performed.

The difference between each consecutive pair of hybrid games is denoted underlined.

Hybrid $\mathbf{H}_0$. The hybrid $\mathbf{H}_0$ is equivalent of the experiment $\mathbf{Exp}^{\texttt{Rand-wRCCA}}$ but the hybrid aborts whenever it finds a collision either in the random oracle $\mathcal{G}$ or in the random oracle $\mathcal{H}$.

Lemma 14. *The Rand-wRCCA experiment on $\mathcal{PKE}$ and $\mathbf{H}_0$ are statistically close.*

Proof. The event that a collision is found in $\mathcal{G}$ or $\mathcal{H}$ is negligible in the security parameter. The two distributions diverge only when such event happens, by the Shoup's difference lemma (lemma 1) the two experiments are statistically close.

Hybrid $\mathbf{H}_1$. In this experiment $\mathbf{H}_1$ the challenge ciphertext is computed as in experiment $\mathbf{H}_0$ except that $\underline{[\mathbf{y}^*] \leftarrow_\$ \mathbb{G}^2 \setminus \mathit{Span}([\mathbf{g}])}$ and $\underline{[\mathbf{w}^*] \leftarrow_\$ \mathbb{G}^2 \setminus \mathit{Span}([\mathbf{g}])}$. Moreover, the ciphertext $\mathtt{C}^* = (\bar{W}^*, \bar{X}^*, \bar{Y}^*)$ is computed using the secret material as follow:

$$[c_w^*] \leftarrow \begin{cases} \mathbf{a}^T \cdot [\mathbf{w}^*] + R^* & \text{if } b^* = 0 \\ \underline{\mathbf{a}^T \cdot [\mathbf{w}^*]} + [\hat{c}_w] & \text{else} \end{cases}, \quad \bar{Z}^* \leftarrow \underline{\bar{F} \cdot [\mathbf{w}^*]}$$

$$\bar{X}^* \leftarrow \begin{cases} x^* \cdot [\mathbf{g}, \mathbf{b}, p^*c + d] + [\mathbf{0}, \bar{\mathtt{M}}, 0] + \bar{Z}^* & \text{if } b^* = 0 \\ \hat{X} + \bar{Z}^* + s \cdot \hat{Y} & \text{else} \end{cases}$$

$$\bar{Y}^* \leftarrow \begin{cases} \underline{[\mathbf{y}^*, \mathbf{B} \cdot \mathbf{y}^*, (p^*\mathbf{c} + \mathbf{d})^T \cdot \mathbf{y}^*]} & \text{if } b^* = 0 \\ t \cdot \hat{Y} & \text{else} \end{cases}$$

where $R^* \leftarrow_\$ \mathbb{G}$, $x^* \leftarrow_\$ \mathbb{Z}_q$, $p \leftarrow \mathcal{G}(\bar{\mathtt{M}} \| \mathcal{H}(R^*))$, $\hat{\mathtt{C}} = (\hat{W}, \hat{X}, \hat{Y})$ is the ciphertext provided by the adversary and $\bar{\mathtt{M}} \leftarrow \mathsf{Dec}(\mathsf{sk}, \hat{\mathtt{C}})$. Moreover, $\underline{\text{we define } \mu_w := \mathbf{a}^T \cdot \mathbf{w}^*}$.

Lemma 15. *If the DDH Assumption holds over $\mathbb{G}$ as generated by* Setup *then $\mathbf{H}_0$ and $\mathbf{H}_1$ are indistinghuishable.*

Easy lemma, proof omitted.

Hybrid $\mathbf{H}_2$. Let experiment $\mathbf{H}_2$ be the same as $\mathbf{H}_1$ but where the decryption oracle executes the following decryption procedure:

$\mathsf{Dec}_2(\mathsf{sk}, \mathtt{C})$: Parse $\mathtt{C} = \bar{W}, \bar{X}, \bar{Y}$ and $\bar{W} = [\mathbf{w}, c_w]$
Compute $\bar{R} \leftarrow [c_w] - \mathbf{a}^T \cdot [\mathbf{w}]$
$\underline{\text{if } \bar{R} \notin \mathcal{Q}^{\mathcal{H}} \text{ then output } \perp,}$
else continue as the decryption algorithm Dec.

Lemma 16. *The hybrids $\mathbf{H}_2$ and $\mathbf{H}_1$ are statistically indistinguishable.*

Proof (Sketch). Let `Decrypt` the event that for a decryption query $\mathtt{C}$ it happens that $\bar{R} \notin \mathcal{Q}^{\mathcal{H}}$ but the decryption oracle in $\mathbf{H}_1$ does not output $\perp$. By the Shoup's difference lemma (see lemma 1), it is easy to see that the statistical distance between the two experiments is bounded by the probability of this event.

Notice that if $\bar{R} \notin \mathcal{Q}^{\mathcal{H}}$ and $\bar{R} \neq \bar{R}^*$ then the value $p \leftarrow \mathcal{G}(\bar{\mathtt{M}} \| \mathcal{H}(\bar{R}))$ is statistically close to be uniformly random over $\mathbb{Z}_q$, and therefore the probability that the decryption equations hold is less or equal than $1/q$. On the other hand if $\bar{R} = \bar{R}^*$, either the $p = p^*$ and then independently on the fact that the decryption equations hold or not the decryption oracle of $\mathbf{H}_1$ would output $\perp$ (or abort if a collision is found) or $p \neq p^*$ and in this case p is statistically close from being uniformly random and again the decryption equations would hold only with negligible probability.

Hybrid $\mathbf{H}_3$. Let experiment $\mathbf{H}_3$ be the same as $\mathbf{H}_2$ but that $\underline{\text{samples } \mu_w \leftarrow_\$ \mathbb{Z}_q \text{ and } \bar{Z}^* \leftarrow_\$ \mathbb{G}^{\ell+3}}$ and the decryption oracle executes the following decryption procedure:

$\mathsf{Dec}_3(\mathsf{sk}, \mathtt{C})$: Parse $\mathtt{C} = \bar{W}, \bar{X}, \bar{Y}$ and $\bar{W} = [\mathbf{w}, c_w]$,
find α_w, β_w such that $\mathbf{w} = \alpha_w \cdot [\mathbf{w}^*] + \beta_w[\mathbf{g}]$,
$\bar{R} \leftarrow [c_w] - (\alpha_w \cdot [\mu_w] + \beta_w \cdot [a])$ and $\bar{X}' \leftarrow \bar{X} - (\alpha_w \cdot \bar{Z}^* + \beta_w \cdot [\mathbf{f}])$,
if $\bar{R} \notin \mathcal{Q}^{\mathcal{H}}$ then output $\perp$,
continue as the decryption algorithm Dec.

Lemma 17. *The hybrids* $\mathbf{H}_3$ *and* $\mathbf{H}_2$ *are equivalently distributed.*

Proof. We use the non-adaptive smoothness (lemma 1) for both the hash function defined by $\mathbf{a}$ and by $\mathbf{F}$. Notice that, by linearity of the hash functions, we compute the randomizer $\bar{R}$ and $\bar{X}'$ equivalently in both hybrids.

Hybrid $\mathbf{H}_4$. Let experiment $\mathbf{H}_4$ be the same as $\mathbf{H}_3$ but where the decryption oracle executes the following decryption procedure:

$\mathsf{Dec}_4(\mathsf{sk}, \mathtt{C})$: Parse $\mathtt{C} = \bar{W}, \bar{X}, \bar{Y}$ and $\bar{W} = [\mathbf{w}, c_w]$,
find α_w, β_w such that $\mathbf{w} = \alpha_w \cdot [\mathbf{w}^*] + \beta_w[\mathbf{g}]$,
$\bar{R} \leftarrow [c_w] - (\alpha_w \cdot [\mu_w] + \beta_w \cdot [a])$ and $\bar{X}' \leftarrow \bar{X} - (\alpha_w \cdot \bar{Z}^* + \beta_w \cdot [\mathbf{f}])$,
if $\alpha_w \neq 0$ and $\bar{R} \neq \hat{R}$ then output $\perp$,
if $\bar{R} \notin \mathcal{Q}^{\mathcal{H}}$ then output $\perp$,
else continue as the decryption algorithm Dec.

In the above $\hat{R}$ is the randomizer computed in the decryption of the ciphertext $\hat{\mathtt{C}}$ chosen by the adversary. (Recall that in the weak re-rand experiment the adversary chooses a ciphertext $\hat{\mathtt{C}}$ that sends to the challenger.)

Lemma 18. *The hybrids* $\mathbf{H}_4$ *and* $\mathbf{H}_3$ *are statistically indistinguishable.*

Proof (Sketch). The proof of the lemma follows closely the proof of Lemma 7. In particular, let $\mathcal{Q}^{\mathsf{Dec}}$ be the queries to the decryption oracle we define the event:

- `InvQuery`: there exists a ciphertext $(\bar{W} = [\mathbf{w}, c_w], \bar{X}, \bar{Y}) \in \mathcal{Q}^{\mathsf{Dec}}$ such that $\alpha_w \neq 0$ and $\bar{R} := ([c_w] - \mathbf{a}^T \cdot [\mathbf{w}]) \in \hat{\mathcal{Q}}^{\mathcal{H}} \setminus \{\hat{R}\}$.

Claim 9 $|\Pr[\mathbf{H}_3] - \Pr[\mathbf{H}_2]| \leq \Pr[\mathtt{InvQuery}]$.

Proof. To prove the claim we rely Shoup's difference lemma (see lemma 1). In fact, conditioned on $\neg\mathtt{InvQuery}$ all queries are such that if β_w such that $\mathbf{w} = \beta_w \cdot \mathbf{g}$ does not exist then $\bar{R} \notin \mathcal{Q}^{\mathcal{H}}$, so for those queries where the predicate holds both $\mathbf{H}_2$ and $\mathbf{H}_3$ would output $\perp$.

We recall that in Lemma 7 we proved that the event happen only with negligible probability (the event is called $\mathtt{InvQuery}_{(a)}$). (In this hybrid experiment and the hybrid experiment of Lemma 7 we sample $\mathbf{w}^*$ is the same way.)

Hybrid $\mathbf{H}_5$. Let experiment $\mathbf{H}_5$ be the same as $\mathbf{H}_4$ but where the decryption oracle executes the following decryption procedure:

$\mathsf{Dec}_5(\mathsf{sk}, \mathtt{C})$: Parse $\mathtt{C} = \bar{W}, \bar{X}, \bar{Y}$ and $\bar{W} = [\mathbf{w}, c_w]$,
find α_w, β_w such that $\mathbf{w} = \alpha_w \cdot [\mathbf{w}^*] + \beta_w[\mathbf{g}]$,
$\bar{R} \leftarrow [c_w] - (\alpha_w \cdot [\mu_w] + \beta_w \cdot [a])$ and $\bar{X}' \leftarrow \bar{X} - (\alpha_w \cdot \bar{Z}^* + \beta_w \cdot [\mathbf{f}])$,
if $\alpha_w \neq 0$ and $\bar{R} \neq \hat{R}$ then output $\perp$,
if $\alpha_w \neq 0$ and $\bar{R} = \hat{R}$ then output $\perp$,
if $\bar{R} \notin \mathcal{Q}^{\mathcal{H}}$ then output $\perp$,
else continue as the decryption algorithm Dec.

Lemma 19. *The hybrids* $\mathbf{H}_5$ *and* $\mathbf{H}_4$ *are statistically indistinguishable.*

Proof. First notice that if $\bar{\mathtt{M}} = \hat{\mathtt{M}}$ then both decryption oracles will always output $\perp$, independently of the validity of the decryption equations, so we consider $\bar{\mathtt{M}} \neq \hat{\mathtt{M}}$. Let $\mathtt{Invalid}$ be the event that the adversary queries with a ciphertext $\mathtt{C}$ such that the randomizer $\bar{R} = \hat{R}$ but for which the decryption procedure Dec_4 would not output $\perp$.

The two hybrids are equivalent if such event does not happen, so applying the Shoup's difference lemma, we need only to show that the event happens with negligible probability. Let $\mathtt{Invalid}_i$ be the event that the adversary queries for the first time with a ciphertext $\mathtt{C}$ such that the randomizer $\bar{R} = \hat{R}$ but for which the decryption procedure Dec_4 would not output $\perp$ at the i-th decryption oracle query. It is easy to check that $\Pr[\mathtt{Invalid}] \leq \sum_i \Pr[\mathtt{Invalid}_i]$.

Claim 10 $\Pr[\mathtt{Invalid}_i] \in \mathtt{negl}(\lambda)$.

Proof. Let **View** be the random variable that describes the view of the adversary up to the i-th decryption oracle query, and let $\bar{Z}^* = (z_1, \ldots, z_{\ell+3})$. We notice that, conditioning on **View**, if $b^* = 0$ then the variable $z_{\ell+3}$ has still q possible ways to be assigned. In fact, the view **View** contains the ciphertext $\mathtt{C}^*$ where x^* is hidden information theoretically by the elements $z_1, \ldots, z_{\ell+2}$ in all the positions, and x^* hides $z_{\ell+3}$ in the last position. Moreover, all the decryption oracle calls with $\alpha_w = 0$ are independent of $\mathbf{Z}^*$, and all the decryption oracle calls with $\alpha_w \neq 0$ were answered with $\perp$ (cause of the definition of the event $\mathtt{Invalid}_i$). Similarly, when $b^* = 1$ the variable $z_{\ell+3}$ has still 2^λ possible ways to be assigned. In fact, in this case the ciphertext would contain the sum $(\hat{s} \cdot p_y + p_x + z_{\ell+3})$ and $\hat{s}$ is uniformly distributed over $\mathbb{Z}_{2^\lambda}$. (As otherwise we could compute $z_{\ell+3}$ with probability better than $2^{-\lambda}$.) So by taking the minimum, the variable $z_{3+\ell}$ has at least 2^λ possible ways to be assigned.

On the other hand, when the event $\mathtt{Invalid}$ happens then the adversary submits a ciphertext $\mathtt{C} = \bar{W}, \bar{X}, \bar{Y}$, where $\bar{X} = [\mathbf{x}, \mathbf{c}_x, p_x]$, for which the decryption equation below must hold:

$$p_x - (\beta_w \cdot f_{\ell+3} + \alpha_w \cdot z_{\ell+3}) = (\mathbf{c} \cdot p + \mathbf{d})^T \cdot \mathbf{x}',$$

and $\mathbf{x}' = \mathbf{x} - (\beta_w \cdot \mathbf{g} + \alpha_w \cdot (z_1, z_2))$. Notice that this equation, when $p \neq \hat{p}$ (cause $\bar{\mathtt{M}} \neq \hat{\mathtt{M}}$) and $\alpha_w \neq 0$, holds with probability at most $2^{-\lambda}$.

Hybrid $\mathbf{H}_6$. Let experiment $\mathbf{H}_6$ be the same as $\mathbf{H}_5$ but where the decryption oracle executes the following decryption procedure:

$\underline{\mathsf{Dec}_6(\mathsf{sk}, \mathtt{C}):}$
Parse $\mathtt{C} = \bar{W}, \bar{X}, \bar{Y}$, $\bar{W} = [\mathbf{w}, c_w]$ and $\bar{Y} = [\mathbf{y}, \mathbf{c}_y, p_y]$
find α_w, β_w such that $\mathbf{w} = \alpha_w \cdot [\mathbf{w}^*] + \beta_w[\mathbf{g}]$,
$\bar{R} \leftarrow [c_w] - (\alpha_w \cdot [\mu_w] + \beta_w \cdot [a])$ and $\bar{X}' \leftarrow \bar{X} - (\alpha_w \cdot \bar{Z}^* + \beta_w \cdot [\mathbf{f}])$,
if $\alpha_w \neq 0$ then output $\perp$,
if $\bar{R} \notin \mathcal{Q}^{\mathcal{H}}$ then output $\perp$,
$\underline{\text{if } \mathbf{x}' \notin \mathit{Span}(\mathbf{g}) \text{ or } \mathbf{y} \notin \mathit{Span}(\mathbf{g}) \text{ then output } \perp.}$
else continue as the decryption algorithm Dec.

Lemma 20. *The hybrids* $\mathbf{H}_6$ *and* $\mathbf{H}_5$ *are statistically indistinguishable.*

Proof (Proof Sketch.). The lemma follows by the standard Cramer-Shoup's analysis over $\mathbf{c}, \mathbf{d}$. We can assume that for all the decryption queries the value p as computed by the decryption procedure is either different than p^* or if it the same then the computed message $\bar{\mathtt{M}}$ is equal to $\hat{\mathtt{M}}$. In fact the event above implies a collision in $\mathcal{G}$ which can be found only with negligible probability.

Let Decrypt be the first time that decryption oracle of $\mathbf{H}_5$ and $\mathbf{H}_6$ diverge, namely Dec_6 outputs $\perp$ while Dec_5 would not and let $\mathtt{C} = (\bar{W}, \bar{X}, \bar{Y})$ be the queried ciphertext.

First notice, whenever the decryption computes $p = p^*$, independently of the validity of the decryption equations, the output of the oracle is $\perp$. Moreover, for each of the decryption query before this where $\mathbf{x}'$ or $\mathbf{y}$ were not in $Span(\mathbf{g})$ both the decryption oracle answered with $\perp$. So given $(p^*\mathbf{c}+\mathbf{d})^T \cdot [\mathbf{y}^*]$ both the values $(p\mathbf{c}+\mathbf{d})^T \cdot [\mathbf{x}']$ and $(p\mathbf{c}+\mathbf{d})^T \cdot [\mathbf{y}]$ are uniformly distributed over a set of size at most $q - |\mathcal{Q}^{\mathsf{Dec}}|$. The latter implies that the decryption equations would hold only with negligible probability.

Hybrid $\mathbf{H}_7$. Let experiment $\mathbf{H}_7$ be the same as $\mathbf{H}_6$ but where the decryption oracle executes the following decryption procedure:

$\underline{\mathsf{Dec}_7(\mathsf{sk}, \mathtt{C})}$:
Parse $\mathtt{C} = \bar{W}, \bar{X}, \bar{Y}$, $\bar{W} = [\mathbf{w}, c_w]$ and $\bar{Y} = [\mathbf{y}, \mathbf{c}_y, p_y]$
find α_w, β_w such that $\mathbf{w} = \alpha_w \cdot [\mathbf{w}^*] + \beta_w[\mathbf{g}]$,
$\bar{R} \leftarrow [c_w] - (\alpha_w \cdot [\mu_w] + \beta_w \cdot [a])$ and $\bar{X}' \leftarrow \bar{X} - (\alpha_w \cdot \bar{Z}^* + \beta_w \cdot [\mathbf{f}])$,
if $\alpha_w \neq 0$ then output $\perp$,
if $\bar{R} \notin \mathcal{Q}^{\mathcal{H}}$ then output $\perp$,
if $\mathbf{x}' \notin Span(\mathbf{g})$ or $\mathbf{y} \notin Span(\mathbf{g})$ then output $\perp$.
Compute p as in Dec and $\underline{\text{if } p \in \{\hat{p}, p^*\}, \text{ then output } \perp}$,
else continue as the decryption algorithm Dec.

Lemma 21. *The hybrids $\mathbf{H}_7$ and $\mathbf{H}_6$ are equivalently distributed.*

Proof. For any oracle query where the event $p \in \{\hat{p}, p^*\}$ happens, let $\bar{\mathtt{M}}'$ be the output of $\mathsf{Dec}_7(\mathsf{sk}, \mathtt{C})$. If $\bar{\mathtt{M}}' = \perp$ or $\bar{\mathtt{M}}' = \bar{\mathtt{M}}$, then there is no difference between $\mathbf{H}_7$ and $\mathbf{H}_6$, as in both cases the oracle answer would be $\perp$. So, the only interesting event that would mark a difference between the two hybrids is that one where $p \in \{\hat{p}, p^*\}$ occurs and $\bar{\mathtt{M}}' \neq \bar{\mathtt{M}}$. However, this event is equal to the event that the adversary finds a collision in the random oracle $\mathcal{G}$, and when this happens both hybrids would abort. Hence, $\mathbf{H}_7 \equiv \mathbf{H}_6$.

Hybrid $\mathbf{H}_8$. Consider the hybrid experiment $\mathbf{H}_8$ be the same as $\mathbf{H}_7$ but where $\bar{X}^*$ is computed as follow:

$$[c_w^*] \leftarrow \begin{cases} [\mu_w] + \underline{\hat{R}} & \text{if } b^* = 0 \\ [\mu_w] + [\hat{c}_w] & \text{else} \end{cases},$$

$$\bar{X}^* \leftarrow \begin{cases} x^* \cdot [\mathbf{g}, \mathbf{b}, \underline{\hat{p} \cdot c + d}] + [\mathbf{0}, \bar{\mathtt{M}}, 0] + \bar{Z}^* & \text{if } b^* = 0 \\ \hat{X} + \bar{Z}^* + s \cdot \hat{Y} & \text{else} \end{cases}$$

$$\bar{Y}^* \leftarrow \begin{cases} [\mathbf{y}^*, \mathbf{B} \cdot \mathbf{y}^*, \underline{(\hat{p} \cdot \mathbf{c} + \mathbf{d})^T} \cdot \mathbf{y}^*] & \text{if } b^* = 0 \\ t \cdot \hat{Y} & \text{else} \end{cases}$$

Lemma 22. *The hybrids $\mathbf{H}_8$ and $\mathbf{H}_7$ are identically distributed.*

Proof. Essentially the changes in $\mathbf{H}_8$ occur only in the case when $b^* = 0$. However, notice that in this case the vector $x^* \cdot [\mathbf{g}, \mathbf{b}, \hat{p}c + d] + [\mathbf{0}, \bar{\mathtt{M}}, 0]$ is masked by the uniformly distributed vector $\bar{Z}^*$; similarly, $\hat{R}$ is masked by the uniformly distributed element $\mathbf{a}^T \cdot [\mathbf{w}^*]$ and both $(p^* \cdot \mathbf{c} + \mathbf{d}) \cdot \mathbf{y}^*$ and $(\hat{p} \cdot \mathbf{c} + \mathbf{d}) \cdot \mathbf{y}^*$ are uniformly distributed. (Recall that we changed the way the decryption oracles are answered so the statements above are true even conditioning on the answers of the decryption oracle.)

In addition, switching from p^* to $\hat{p}$ in $\bar{X}^*$ does not introduce any difference in the answers to decryption queries since, by the change introduced in hybrid $\mathbf{H}_7$, the decryption oracle behaves the same.

Hybrid $\mathbf{H}_9$. Consider the hybrid experiment $\mathbf{H}_9$ be the same as $\mathbf{H}_8$ but where $\bar{Z}^*$ is sampled as follow:

$$\bar{Z}^* \leftarrow_{\$} \begin{cases} -x^* \cdot [\mathbf{g}, \mathbf{b}, \hat{p} \cdot c + d] - [\mathbf{0}, \bar{\mathtt{M}}, 0] + \hat{X} + s \cdot \hat{Y} + \bar{Z}' & \text{if } b^* = 0 \\ \bar{Z}' & \text{else} \end{cases}$$

where $\bar{Z}' \leftarrow_{\$} \mathbb{G}^{\ell+3}$.

Lemma 23. *The hybrids $\mathbf{H}_9$ and $\mathbf{H}_8$ are identically distributed.*

Proof. We first recall that the decryption oracle does not depend on $\bar{Z}^*$, so the only part of the view of the adversary that depends on the random variable $\bar{Z}^*$ is the challenge re-randomized/fresh ciphertext. Moreover, this change is only conceptual as the distribution described above is equivalent to the uniform distribution over $\mathbb{G}^{\ell+3}$. In fact, we notice that $\bar{Z}^*$ in $\mathbf{H}_9$ does not depend on $b^*, x^*, s, \hat{p}, \bar{\mathtt{M}}$. In particular for any assignment of these variables the distribution of $\bar{Z}^*$ is uniformly random over $\mathbb{G}^{\ell+3}$, even given the full view of the adversary.

Before we describe the next hybrid, we notice that $\mathbf{H}_9$ computes the challenge ciphertext as follow:

$$\begin{aligned} \bar{W}^* &\leftarrow \begin{cases} [\mathbf{w}^*, \mu_w] + ([0], [0], \hat{R}) & \text{if } b^* = 0 \\ [\mathbf{w}^*, \mu_w] + \hat{W} & \text{else} \end{cases}, \\ \bar{X}^* &\leftarrow \hat{X} + s \cdot \hat{Y} + \bar{Z}', \\ \bar{Y}^* &\leftarrow \begin{cases} [\mathbf{y}^*, \mathbf{B} \cdot \mathbf{y}^*, (\hat{p} \cdot \mathbf{c} + \mathbf{d})^T \cdot \mathbf{y}^*] & \text{if } b^* = 0 \\ t \cdot \hat{Y} & \text{else} \end{cases} \end{aligned}$$

Hybrid $\mathbf{H}_{10}$. We revert the changes introduced in $\mathbf{H}_3, \mathbf{H}_4, \mathbf{H}_5$ and $\mathbf{H}_6$ (which are not efficiently computable).

Lemma 24. *The hybrids $\mathbf{H}_{10}$ and $\mathbf{H}_9$ are statistically close.*

The lemma follows applying the same arguments of the previous lemmas.

Hybrid $\mathbf{H}_{11}$. We revert the change introduced in $\mathbf{H}_1$. Specifically, we sample $\mathbf{y}^*$ from the distribution $Span([\mathbf{g}])$.

Lemma 25. *The hybrids $\mathbf{H}_{11}$ and $\mathbf{H}_{10}$ are computationally indistinguishable.*

Notice that the hybrid $\mathbf{H}_{10}$ can be computed efficiently, we simply apply the DDH Assumption. The proof of the lemma is trivial, therefore omitted.

We end the proof of the theorem noticing that in $\mathbf{H}_{11}$ the distribution of $\mathtt{C}^*$ is identical in both cases when $b^* = 0$ and $b^* = 1$. Therefore $\Pr[\mathbf{H}_{11} = 1] = \frac{1}{2}$. The theorem follows by the lemmas and the triangle inequality.

4.6 Proof of Theorem 5 (unforgeability)

Proof. To prove the theorem we describe a sequences of hybrids. The difference between each consecutive pair of hybrid games is denoted underlined.

Similarly to the proof of Theorem 2 and Theorem 3 we consider a decryption procedure that always fixes the randomizer t to 0. We can do this without loss of generality for the same reasons of proof of Theorem 3, namely, the randomized part of the decryption procedure can be publicly performed and a valid forged ciphertext $\mathtt{C}^*$ (see Fig. 5) of the UF-CCA security experiment over the original encryption scheme is also a valid forged ciphertext for the modified encryption scheme.

Hybrid H'. Consider the hybrid experiment $\mathbf{H}'$ be the same as $\mathbf{Exp}^{\mathtt{UF-CCA}}$ but where all the ciphertexts $\mathtt{C}_i$ for $i \in [n]$ are rerandomizated using the private hashing procedure. Specifically, for all $i \in [n]$ the ciphertext $\mathtt{C}'_i = (\bar{W}'_i, \bar{X}'_i, \bar{Y}'_i)$ is computed as follow:

$$\begin{aligned}
&\bar{W}'_i \leftarrow \kappa \cdot \bar{W}_i + [\mathbf{w}'_i, \underline{\mathbf{a}^T \cdot \mathbf{w'}_i}] && \text{where } \mathbf{w}'_i = w'_i \cdot \mathbf{g},\\
&\bar{X}'_i \leftarrow \kappa \cdot \bar{X}_i + s \cdot \bar{Y}_i + \underline{\mathbf{F} \cdot [\mathbf{w}'_i]}\\
&\bar{Y}'_i \leftarrow \kappa \cdot t \cdot \bar{Y}_i
\end{aligned}$$

Moreover, the hybrid $\underline{\text{sample the values } \kappa'_1, \ldots, \kappa'_n \leftarrow_\$ \mathbb{Z}_q}$.

Lemma 26. *The hybrids* $\mathbf{H}'$ *and the experiment* $\mathbf{Exp}^{\mathtt{UF-CCA}}$ *are equivalent.*

This follows from the projective hashing property, namely $\mathbf{a}^T \cdot [\mathbf{w}'_i] = w'_i \cdot [a]$.

Next, we consider a series of hybrids indexed by $i = 1, \ldots, n$.

Hybrid $\mathbf{H}_i$. This hybrid $\mathbf{H}_i$ is the experiment that runs the same as $\mathbf{H}'$ but for $j = 1$ to $i-1$, it re-randomizes the j-th ciphertext using κ'_j as re-randomization key, instead of κ. Specifically, for all $j \geq i$, $\mathtt{C}'_j = (\bar{W}'_j, \bar{X}'_j, \bar{Y}'_j)$ is computed as in $\mathbf{H}'$, while $\underline{\text{for } j < i \text{ the ciphertext } \mathtt{C}'_j \text{ is computed as follows}}$:

$$\begin{aligned}
&\bar{W}'_i \leftarrow \underline{\kappa'_j} \cdot \bar{W}_i + [\mathbf{w}'_i, \mathbf{a}^T \cdot \mathbf{w'}_i] && \text{where } \mathbf{w}'_i = w'_i \cdot \mathbf{g},\\
&\bar{X}'_i \leftarrow \underline{\kappa'_j} \cdot \bar{X}_i + s \cdot \bar{Y}_i + \mathbf{F} \cdot [\mathbf{w}'_i]\\
&\bar{Y}'_i \leftarrow \underline{\kappa'_j} \cdot t \cdot \bar{Y}_i
\end{aligned}$$

It is easy to see that our hybrid $\mathbf{H}'$ described above is the same as hybrid $\mathbf{H}_1$ where all the ciphertexts are re-randomized using the same κ.

To prove the theorem we show that each consecutive pair is computationally indistinguishable, i.e., $\mathbf{H}_i \approx_c \mathbf{H}_{i+1}$. To this end (and for ease of analysis), for every $1 \leq i \leq n$, we argue the transition from hybrid i to $i+1$ using other 7 intermediate experiments.

Namely, we show:

$$\mathbf{H}_1 \approx_c \mathbf{H}_{1,1} \equiv \cdots \equiv \mathbf{H}_{1,7} \approx_c \mathbf{H}_2 \approx_c \cdots \approx_c \mathbf{H}_n \equiv \mathbf{H}_{n,1} \equiv \cdots \equiv \mathbf{H}_{n,7} \approx_c \mathbf{H}_{n+1}$$

and finally we will argue that the probability of winning in $\mathbf{H}_{n+1}$ is negligible.

Hybrid i,1 Consider the hybrid experiment $\mathbf{H}_{i,1}$ be the same as $\mathbf{H}_i$ but where the i-th ciphertext is rerandomized using $\mathbf{w}'_i$ and $\mathbf{y}_i$ taken from a different distribution. Specifically, the i-th rerandomizated ciphertext $\mathtt{C}'_i = (\bar{W}'_i, \bar{X}'_i, \bar{Y}'_i)$ is computed as follow:

$$\begin{aligned}
&\bar{W}'_i \leftarrow \kappa \cdot \bar{W}_i + [\mathbf{w}'_i, \mathbf{a}^T \cdot \mathbf{w'}_i] && \text{where } \underline{\mathbf{w}'_i \leftarrow_\$ \mathbb{G}^2}\\
&\bar{X}'_i \leftarrow \kappa \cdot \bar{X}_i + s \cdot \bar{Y}_i + \mathbf{F} \cdot [\mathbf{w}'_i]\\
&\bar{Y}'_i \leftarrow \kappa \cdot t \cdot \bar{Y}_i
\end{aligned}$$

Lemma 27. *If the DDH Assumption holds over $\mathbb{G}$ as generated by* Setup *then* $\mathbf{H}_i \approx_c \mathbf{H}_{i,1}$.

Easy lemma, proof omitted.

Hybrid i,2 Let the hybrid experiment $\mathbf{H}_{i,2}$ be the same as $\mathbf{H}_{i,1}$ except that the experiment uses the following decryption procedure $\mathsf{ADec}_{i,2}$ instead of ADec.

On input all the secrets of the experiment (notably including sk and $\mathbf{w}'_i, \mathbf{y}'_i$), a re-randomization key κ and any ciphertext $\mathtt{C} = (\bar{W}, \bar{X}, \bar{Y})$, $\mathsf{ADec}_{i,2}$ proceeds the same as ADec except that values $R, \bar{X}'$ are computed as follows:

Find $\alpha_w, \beta_w \in \mathbb{Z}_q$ such that $\mathbf{w} = \alpha_w \cdot \mathbf{w}'_i + \beta_w \cdot \mathbf{g}$.

$$R \leftarrow ([c_{w,j}] - \alpha_w \cdot [\mathbf{a}^T \cdot \mathbf{w}'_i] - \beta_w \cdot [a])/\kappa$$
$$\bar{X}' \leftarrow \bar{X}_j - \alpha_w \cdot [\mathbf{F} \cdot \mathbf{w}'_i] - \beta_w \cdot [\mathbf{f}]$$

Lemma 28. *The hybrids* $\mathbf{H}_{i,2}$ *and* $\mathbf{H}_{i,1}$ *are equivalent.*

The lemma follows by looking at the definition of $\mathbf{w}$ and by the definition of $[a] = [\mathbf{a}^T \cdot \mathbf{w}]$ and $[\mathbf{f}] = [\mathbf{F} \cdot \mathbf{g}]$.

Hybrid i,3 Let the hybrid experiment $\mathbf{H}_{i,3}$ be the same as $\mathbf{H}_{i,2}$ except that:

1. it defines two new variables: $\mu_{w,i} = \mathbf{a}^T \cdot \mathbf{w}'_i \in \mathbb{Z}_q$ and $\bar{Z}_i = \mathbf{F} \cdot \mathbf{y}'_i \in \mathbb{Z}_q^{\ell+3}$;
2. it uses $\mu_{w,i}, \bar{Z}_i$ to re-reandomize the i-th ciphertext $\mathtt{C}_i = (\bar{W}_i, \bar{X}_i, \bar{Y}_i)$ as follows:

$$\bar{W}'_i \leftarrow \kappa \cdot \bar{W}_i + [\mathbf{w}'_i, \mu_{w,i}] \qquad \text{where } \mathbf{w}'_i \leftarrow_\$ \mathbb{G}^2$$
$$\bar{X}'_i \leftarrow \kappa \cdot \bar{X}_i + s \cdot \bar{Y}_i + \bar{Z}_i$$
$$\bar{Y}'_i \leftarrow \kappa \cdot t \cdot \bar{Y}_i$$

3. it uses a decryption algorithm $\mathsf{ADec}_{i,3}$ that is the same as $\mathsf{ADec}_{i,2}$ except for:

$$R \leftarrow ([c_{w,j}] - \alpha_w \cdot [\mu_{w,i}] - \beta_w \cdot [a])/\kappa$$
$$\bar{X}' \leftarrow \bar{X}_j - \alpha_y \cdot [\bar{Z}_i] - \beta_y \cdot [\mathbf{f}]$$

Lemma 29. *The hybrids* $\mathbf{H}_{i,2}$ *and* $\mathbf{H}_{i,3}$ *are equivalent.*

This is only a syntactic change, introduced for ease of exposition. Thus the lemma is obvious.

Hybrid i,4 Let the hybrid experiment $\mathbf{H}_{i,4}$ be the same as $\mathbf{H}_{i,3}$ except that it samples $\mu_{w,i} \leftarrow_\$ \mathbb{Z}_q$ and $\bar{Z}_i \leftarrow_\$ \mathbb{Z}_q^{m+3}$.

Lemma 30. *The hybrids* $\mathbf{H}_{i,3}$ *and* $\mathbf{H}_{i,4}$ *are equivalent.*

The lemma follows by applying Corollary 1. Slightly more in detail, we rely on that given $(\mathbf{a}^T \cdot \mathbf{g}, \mathbf{F} \cdot \mathbf{h}, \mathbf{w}'_i, \mathbf{w}'_i)$, the values $\mathbf{a}^T \cdot \mathbf{w}'_i$ and $\mathbf{F} \cdot \mathbf{w}'_i$ are identically distributed to random elements in $\mathbb{Z}_q$ and $\mathbb{Z}_q^{\ell+3}$ respectively. In particular, notice that to apply Corollary 1 we need the hybrids to be computable without knowing the secret keys $\mathbf{a}^T, \mathbf{F}$: this is possible due to the changes introduced in $\mathbf{H}_{i,2}, \mathbf{H}_{i,3}$.

Hybrid i,5 Let the hybrid experiment $\mathbf{H}_{i,5}$ be the same as $\mathbf{H}_{i,4}$ except that the i-th ciphertext $\mathtt{C}_i = (\bar{W}_i, \bar{X}_i, \bar{Y}_i)$ is re-randomized by using κ'_i instead of κ. Namely:

$$\bar{W}'_i \leftarrow \kappa'_i \cdot \bar{W}_i + [\mathbf{w}'_i, \mu_{w,i}] \qquad \text{where } [\mathbf{w}'_i, \mu_{w,i}] \leftarrow_\$ \mathbb{G}^3$$
$$\bar{X}'_i \leftarrow \kappa'_i \cdot \bar{X}_i + s \cdot \bar{Y}_i + \bar{Z}_i \qquad \text{where } \bar{Z}_i \leftarrow_\$ \mathbb{G}^{\ell+3}$$
$$\bar{Y}'_i \leftarrow \kappa'_i \cdot t \cdot \bar{Y}_i$$

Lemma 31. *The hybrids* $\mathbf{H}_{i,4}$ *and* $\mathbf{H}_{i,5}$ *are equivalent.*

First we notice that for the component $\bar{Y}'_i$ this is a simple rewriting, infact t could be sampled as $t' \cdot \frac{\kappa'_i}{\kappa}$ where $t' \leftarrow \mathbb{Z}_q$. For the remaining components, as one can see, in these hybrids the vectors $\bar{W}'_i, \bar{X}'_i$ are essentially chosen uniformly and independently at random. In other words, the random masks $[\mathbf{w}'_i, \mu_{w,i}] \leftarrow_\$ \mathbb{G}^3$, $\bar{Z}_i \leftarrow_\$ \mathbb{G}^{\ell+3}$ act as a one-time pad and thus switching κ into κ'_i can be done by keeping the distribution of these elements identical.

In the following two hybrids we essentially revert all the changes made from $\mathbf{H}_{i,4}$ back to $\mathbf{H}_{i,1}$.

Hybrid i,6 Let the hybrid experiment $\mathbf{H}_{i,6}$ be the same as $\mathbf{H}_{i,5}$ except that it defines $\mu_{w,i} = \mathbf{a}^T \cdot \mathbf{w}'_i \in \mathbb{Z}_q$ and $\bar{Z}_i = \mathbf{F} \cdot \mathbf{y}'_i \in \mathbb{Z}_q^{m+3}$.

Hybrid i,7 Let the hybrid experiment $\mathbf{H}_{i,7}$ be the same as $\mathbf{H}_{i,6}$ except that it uses the decryption algorithm ADec.

Lemma 32. *The hybrids* $\mathbf{H}_{i,5}$, $\mathbf{H}_{i,6}$ *and* $\mathbf{H}_{i,6}$ *are identically distributed.*

$\mathbf{H}_{i,5} \equiv \mathbf{H}_{i,6}$ is justified by the same argument used to argue $\mathbf{H}_{i,3} \equiv \mathbf{H}_{i,4}$. $\mathbf{H}_{i,6} \equiv \mathbf{H}_{i,7}$ is justified by the same arguments used to argue $\mathbf{H}_{i,2} \equiv \mathbf{H}_{i,1}$.

Finally, we recall hybrid $\mathbf{H}_{i+1}$ where the i-th re-randomized ciphertext given to the adversary is a correctly distributed re-randomization but done using a random and independent re-randomization key κ'_i.

Hybrid i+1 Let the hybrid experiment $\mathbf{H}_{i+1}$ be the same as $\mathbf{H}_{i,7}$ except that it samples $\mathbf{w}'_i \leftarrow_\$ \mathit{Span}(\mathbf{g})$.

Lemma 33. *If the DDH assumption holds over* $\mathbb{G}$ *as generated by* Setup, $\mathbf{H}_{i+1} \approx_c \mathbf{H}_{i,7}$.

The proof is the same as that used to argue $\mathbf{H}_0 \approx_c \mathbf{H}_{i,1}$.

Finally, we show that in the last hybrid $\mathbf{H}_{n+1}$ the adversary has only negligible probability of winning.

Lemma 34. *There exist a negligible function* $\mathtt{negl}$ *such that* $\Pr[\mathbf{H}_{n+1} = 1] \leq \mathtt{negl}(\lambda)$

Proof. Notice that in the hybrid $\mathbf{H}_{n+1}$ the re-randomization key κ used to decrypt the ciphertext $\mathtt{C}^*$ does not appear anymore in the view of the adversary. We argue that over the random choice of $\kappa \leftarrow_\$ \mathbb{Z}^*_{2^\lambda}$, $\mathsf{ADec}(\mathsf{sk}, \tilde{\kappa} \cdot \kappa, \mathtt{C}^*) \neq \perp$ occurs only with negligible probability.

To see this, we observe that a random κ makes the value $R^* = ([c^*_w] - \mathbf{a}^T \cdot [\mathbf{w}^*])/\tilde{\kappa} \cdot \kappa$ uniformly distributed over a set of $2^\lambda - 1$ elements (recall that we check for $[c^*_w] - \mathbf{a}^T \cdot [\mathbf{w}^*] \neq 0$). Hence, we have that, in a cascade fashion, the value $p^* = \mathcal{G}(\mathtt{M}^* || \mathcal{H}(R^*))$ is also uniformly distributed, and this makes the equation $[p^*_x] = (p^* \cdot \mathbf{c} + \mathbf{d})^T \cdot \mathbf{x}^*$ (similarly, $[p^*_y] = (p^* \cdot \mathbf{c} + \mathbf{d})^T \cdot \mathbf{y}^*$) satisfiable with probability at most $1/q$, which is negligible.

The proof of the theorem follows by joining together all the lemmas thanks to the triangular inequality.

Remark 2 (On the sufficient assumptions over the hash function $\mathcal{G}$). Lemma 10 can be proved based only on collision resistance of $\mathcal{G}$ cause we need only $p \neq p^*$, Lemma 11 (resp. lemma 12) can be proved based only on the property that $\mathcal{G}(\bar{\mathtt{M}} \| \mathtt{ak}^*)$ (resp. $\mathcal{G}(\bar{\mathtt{M}}_{1-b^*} \| \mathtt{ak}^*)$) is uniformly distributed for any $\bar{\mathtt{M}}$, i.e., it is sufficient that for any assignment of $\bar{\mathtt{M}}$ the function is regular. Lemma 18 can be proved based only on collision resistance of $\mathcal{G}$ cause we need only $p \notin \{\hat{p}, \neq p^*\}$, in the proof of Theorem 5 we do not use any property of the hash functions.

5 Our Optimistic Mixing Construction

In this section we describe a protocol that realizes the optimistic mixing functionality of Fig. 3. The main building blocks of our protocol are the PKE scheme in the previous section, and two NIZK proof systems for proving statements about the ciphertexts of our PKE scheme (see Appendix A) . Our protocol is given in an hybrid world with a non-programmable random oracle and an ideal functionality $\mathcal{F}^{\mathsf{Dec}}$ (see Fig. 6) that models the key generation and decryption of the PKE, and the common reference string generation of the NIZKs. A protocol for realizing the initialization phase of $\mathcal{F}^{\mathsf{Dec}}$ can be obtained using fairly standard techniques for distributed key/crs generation. On the other hand, computing ADec in a distributed way is more tricky. While a realization can be obtained using general-purpose techniques, the realization of an efficient protocol is left for future work.

Building blocks and main ideas. Recall that in our scheme a ciphertext $\mathtt{C}$ is a tuple $(\bar{W}, \bar{X}, \bar{Y})$, where $\bar{W} = [w \cdot \mathbf{g}, w \cdot a + R] \in \mathbb{G}^3$ for random $w \in \mathbb{Z}_q$, and $R \in \mathbb{G}$. Our protocol uses:

- $\mathcal{NIZK}_{\mathtt{sd}} = (\mathsf{Init}_{\mathtt{sd}}, \mathsf{P}_{\mathtt{sd}}, \mathsf{V}_{\mathtt{sd}})$, a NIZK Proof of Knowledge with labels in the CRS+NPRO model for the NP relation

$$\mathcal{R}_{\mathtt{sd}} = \left\{((\bar{S}, \bar{W}), (w, R)) : \ \bar{W} = w \cdot \bar{S} + (0, 0, R)\right\},$$

  where $(\bar{S}, \bar{W}) \in \mathbb{G}^6$ is the instance and $(w, R) \in \mathbb{Z}_q \times \mathbb{G}$ is the witness. For $\mathcal{NIZK}_{\mathtt{sd}}$ we require simulation f-extractability with $f(w, R) = R$.
  This NIZK can be used to prove knowledge of the randomizer R used to create a ciphertext.
- $\mathcal{NIZK}_{\mathtt{mx}} = (\mathsf{Init}_{\mathtt{mx}}, \mathsf{P}_{\mathtt{mx}}, \mathsf{V}_{\mathtt{mx}})$, a NIZK Proof of Membership with labels in the CRS+NPRO model for the NP relation

$$\mathcal{R}_{\mathtt{mx}} = \left\{((\bar{S}, \bar{R}), w) \in \mathbb{G}^6 \times \mathbb{Z}_q : \ \bar{R} = w \cdot \bar{S}\right\},$$

  where $(\bar{S}, \bar{R}) \in \mathbb{G}^6$ is the instance and $w \in \mathbb{Z}_q$ the witness. This NIZK can be used to prove that the $\bar{W}$ component of a ciphertext is a Diffie-Hellman tuple $\bar{W} = w \cdot [\mathbf{g}, a]$ (i.e., it encrypts a neutral randomizer). By homomorphic property, this is also useful to show that two ciphertexts encrypt the same randomizer R.

Given two lists of ciphertexts $\mathcal{L} = \langle \mathtt{C}_1, \ldots, \mathtt{C}_n \rangle$ and $\mathcal{L}' = \langle \mathtt{C}'_1, \ldots, \mathtt{C}'_n \rangle$, and a randomization key $\kappa \in \mathbb{Z}_q$, we define the *checksum* of these lists as the output of the following procedure:

Procedure $\mathbf{CkSum}(\mathcal{L}, \mathcal{L}', \kappa)$:
1. For all $j \in [n]$ parse $\mathtt{C}_j = (\bar{W}_j, \bar{X}_j, \bar{Y}_j)$ and $\mathtt{C}'_j = (\bar{W}'_j, \bar{X}'_j, \bar{Y}'_j)$;
2. Output $\sum_j \bar{W}'_j - \kappa \cdot \sum_j \bar{W}_j$.

The combination of $\mathcal{NIZK}_{\mathtt{mx}}$ and **CkSum** can be used to show that the sum of the randomizers R_j of the ciphertexts in $\mathcal{L}$ is the same as the sum of the randomizers R'_j of the ciphertexts in $\mathcal{L}'$. In particular, this is true when the ciphertexts in $\mathcal{L}'$ are shuffled re-randomizations (with key κ) of those in $\mathcal{L}$. To see this, assume that $\mathcal{L}'$ contains a shuffled re-randomization, with key κ, of the ciphertexts in $\mathcal{L}$, i.e., for every $j \in [n]$, $\bar{W}'_{\pi(j)} \leftarrow \kappa \cdot \bar{W}_j + w'_j \cdot [\mathbf{g}, a]$. Then one can see that $(\sum_j \bar{W}'_j - \kappa \cdot \sum_j \bar{W}_j) \in \mathcal{R}_{\mathtt{mx}}$ with witness $(\sum_j w'_j)$.

A key idea of our optimistic mixing protocol is that proving $\mathbf{CkSum}(\mathcal{L}, \mathcal{L}', \kappa) \in \mathcal{R}_{\mathtt{mx}}$ is "almost" a proof of shuffling. More precisely, combining this NIZK proof with the RCCA security property of our PKE scheme gives us a designated-verifier optimistic proof of shuffling, which can be verified by checking the NIZK, and by decrypting the list $\mathcal{L}'$ and checking that no ciphertext decrypts to $\perp$.

However, since our protocol cannot decrypt every list (as it would reveal the permutations) but only the last one, we have to ensure that if no $\perp$ appears while decrypting the last list, then no $\perp$ would have appeared when decrypting a previous list. This way verifying all the NIZKs and decrypting the last list would be equivalent to ensuring that every mixer correctly shuffled the ciphertexts. We achieve this property thanks to the UF-CCA security of our PKE that, intuitively, guarantees that the adversary cannot repair ciphertexts after an honest authenticated re-randomization.

Functionality $\mathcal{F}^{\mathsf{Dec}}$:

Let $\mathtt{Dec}$ be the name of the functionality.

Initialization Phase: At the first activation do the following:
1. Sample $\mathsf{pk}, \mathsf{sk} \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$ where $\mathsf{pk} = [\mathbf{g}, \mathbf{h}, a, \mathbf{b}, c, d]$;
2. Sample $\mathsf{crs}_{\mathtt{sd}} \leftarrow\!\!\$\ \mathsf{Init}_{\mathtt{sd}}(1^\lambda)$ of the NIZK for $\mathcal{R}_{\mathtt{sd}}$.
3. Sample $\mathsf{crs}_{\mathtt{mx}} \leftarrow\!\!\$\ \mathsf{Init}_{\mathtt{mx}}(1^\lambda)$ of the NIZK for $\mathcal{R}_{\mathtt{mx}}$.
4. Set $\mathsf{pub} \leftarrow (\mathsf{pk}, \mathsf{crs}_{\mathtt{sd}}, \mathsf{crs}_{\mathtt{mx}})$ and $\mathsf{sec} \leftarrow \mathsf{sk}$;

Store the tuple $(\mathsf{pub}, \mathsf{sec})$;

Public Value: On message $\mathtt{pk}$ on any input port or on the influence port send the message pub to the respective output port or on the leakage port and return;

Decode Value: On message $(\mathtt{dec}, \kappa, \mathtt{C})$ on the inport $\mathtt{Dec.in}_{M_i}$ check that the tuple $(\mathtt{C}, \kappa, \mathtt{M}, \mathcal{I})$ exists in the database, if so update $\mathcal{I}$ including the index i else create the new entry $(\mathtt{C}, \kappa, \mathsf{ADec}(\mathsf{sk}, \kappa, \mathtt{C}), \{i\})$ in the database. If $|\mathcal{I}|$ equals m then send $(\mathtt{dec}, X, \kappa, \mathtt{M})$ to the leakage outport;

Deliver: On message $(\mathtt{dec}, \kappa, \mathtt{C}, i)$ on the influence port, If the tuple $(X, \kappa, \mathtt{M}, [m])$ exists in the database then send the message $(\mathtt{dec}, X, \kappa, \mathtt{M})$ in the outport $\mathtt{Dec.out}_{M_i}$.

Functionality $\mathcal{F}^{\mathsf{NPRO}}$:

Let $\mathtt{NPRO}$ the name of the functionality.

Query: Upon message $(\mathtt{query}, x)$ on a protocol's port $\mathtt{NPRO.in}_i$ (resp. on the influence port $\mathtt{NPRO.infl}$) check if (x, y) is in the database, if sample uniformly random $y \leftarrow\!\!\$\ \{0,1\}^\lambda$ and add the entry (x, y) in the database; Send the message y to the outport $\mathtt{NPRO.out}_i$ (resp. to the leakage port $\mathtt{NPRO.lk}$).

Fig. 6: Ideal Functionality for Init and Decode and for the Non-Programmable Random Oracle.

5.1 A high-level description of the protocol

Here we provide a description of our optimistic mixing protocol without going into the full formalism of the UC model. All the parties in the protocol have input $\mathsf{pub} = (\mathsf{pk}, \mathsf{crs}_{\mathtt{sd}}, \mathsf{crs}_{\mathtt{mx}})$, where pk is the public key of the RCCA encryption scheme, and $\mathsf{crs}_{\mathtt{sd}}$ and $\mathsf{crs}_{\mathtt{mx}}$ are the common reference strings for the NIZK proof systems. we write $\mathtt{RO}(x)$ for an invocation of the non-programmable random oracle functionality over input x.

Input Submission. Every sender $\mathcal{P}_{S_j}$, with $j \in [n]$, encrypts its message $\mathtt{M}_j$ by computing $\mathtt{C}_j \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathtt{M}_j)$, and creates a NIZK proof of knowledge $\Pi_j^{\mathtt{s}}$, with label j, of the random values w_j, R_j used to create $\mathtt{C}_j$, i.e., it computes $\Pi_j^{\mathtt{s}} \leftarrow\!\!\$\ \mathsf{P}_{\mathtt{sd}}(\mathsf{crs}_{\mathtt{sd}}, (j, [\mathbf{g}, a], \bar{W}_j), (w_j, R_j))$ where $\bar{W}_j$ is the first component of $\mathtt{C}_j$. Finally, $\mathcal{P}_{S_j}$ posts $(\mathtt{C}_j, \Pi_j^{\mathtt{s}})$ on the bulletin board.

Optimistic Mixing. Once all the senders are done with the previous phase, let $\mathcal{L}_0$ be the list of ciphertexts they posted on the BB such that the NIZK proofs verify. To simplify

the exposition of the result, we assume that all the NIZK proofs $\{\Pi_j^{\mathtt{s}}\}_{j\in[n]}$ verify, so that $\mathcal{L}_0 = \langle \mathtt{C}_{0,j}\rangle_{j\in[n]}$. We can assume this without loss of generality[5].

For $i = 1$ to m, the mixer $\mathcal{P}_{M_i}$ waits for the previous mixer to be done and does the following:

1. Sample a permutation $\pi_i \leftarrow_{\$} \mathcal{S}_n$, and a secret randomization key $\kappa_i \leftarrow_{\$} \mathbb{Z}_q$, and computes $\mathsf{com}_i \leftarrow \mathtt{RO}(\kappa_i)$;
2. Read from the BB the list of ciphertexts $\mathcal{L}_{i-1}$ posted by the previous mixer (or read $\mathcal{L}_0$ if this is the first mixer), and parse $\mathcal{L}_{i-1} = \langle \mathtt{C}_{i-1,j}\rangle_{j\in[n]}$;
3. Build the list $\mathcal{L}_i \leftarrow \langle \mathtt{C}_{i,j}\rangle_{j\in[n]}$ of shuffled and re-randomized ciphertexts by computing
$$\mathtt{C}_{i,\pi_i(j)} \leftarrow \mathsf{ARand}(\mathsf{pk}, \kappa_i, \mathtt{C}_{i-1,j}; w_j, y_j, s_j, t_j),$$
where, for every $j \in [n]$, $w_j, y_j, s_j, t_j \leftarrow_{\$} \mathbb{Z}_q$ are randomly chosen;
4. Compute the value $w_i^* = \sum_{j=1}^{n} w_j$, and store it locally;
5. Post $(\mathcal{L}_i, \mathsf{com}_i)$ on the BB.

Mixing Proof. Once all mixers are done with the previous phase, every mixer $\mathcal{P}_{M_i}$ (this step can be computed concurrently by all the mixers) computes a NIZK proof $\Pi_i^{\mathtt{M}} \leftarrow_{\$} \mathsf{P}_{\mathtt{mx}}(\mathsf{crs}_{\mathtt{mx}}, (i, [\mathbf{g}, a], \mathbf{CkSum}(\mathcal{L}_{i-1}, \mathcal{L}_i, \kappa_i)), w_i^*)$, and posts $(\Pi_i^{\mathtt{M}}, \kappa_i)$ on the BB. With such a message, every mixer $\mathcal{P}_{M_i}$ is opening the commitment com_i, and proving in zero-knowledge that the new list $\mathcal{L}_i$ it produced contains ciphertexts that have the same randomizers as in the previous list $\mathcal{L}_{i-1}$ (precisely, the sum of randomizers $\sum_j R_j$ in the two lists is the same).

Verification. Once all mixers are done with the previous phase, every mixer $\mathcal{P}_{M_i}$ executes the following:

1. Read the messages $(\mathcal{L}_i, \mathsf{com}_i)$ and $(\Pi_i^{\mathtt{M}}, \kappa_i)$ posted by every mixer on the BB, as well as the messages $(\mathtt{C}_{0,j}, \Pi_j^{\mathtt{s}})$ posted by the senders;
2. For all $j \in [n]$ check that $\mathsf{V}_{\mathtt{sd}}(\mathsf{crs}_{\mathtt{sd}}, (j, [\mathbf{g}, a], \bar{W}_j), \Pi_j^{\mathtt{s}}) = 1$, where $\bar{W}_j$ is the first component of ciphertext $\mathtt{C}_{0,j}$;
3. For all $i \in [m]$, check $\mathsf{V}_{\mathtt{mx}}(\mathsf{crs}_{\mathtt{mx}}, (i, [\mathbf{g}, a], \mathbf{CkSum}(\mathcal{L}_{i-1}, \mathcal{L}_i, \kappa_i)), \Pi_i^{\mathtt{M}}) = 1$ and $\mathtt{RO}(\kappa_i) = \mathsf{com}_i$;
4. If all checks verified set $\mathsf{flg}_{\mathtt{valid}} \leftarrow \mathsf{true}$, else set $\mathsf{flg}_{\mathtt{valid}} \leftarrow \mathsf{false}$;
5. Compute $\kappa^* \leftarrow \prod_{i=1}^{m} \kappa_i$;

Decode. All the mixers $\mathcal{P}_{M_i}$ execute the following in parallel (using the ideal functionality $\mathcal{F}^{\mathsf{Dec}}$ to compute decryptions):

1. If $\mathsf{flg}_{\mathtt{valid}} = \mathsf{true}$, let $\mathcal{L}_m = \langle \mathtt{C}_j^*\rangle_{j\in[n]}$ be the list of ciphertexts returned by the last mixer. For $j = 1$ to n, decrypt $\mathtt{M}_j \leftarrow \mathsf{ADec}(\mathsf{sk}, \kappa^*, \mathtt{C}_j^*)$; if $\mathtt{M}_j = \perp$ set $\mathsf{flg}_{\mathtt{valid}} \leftarrow \mathsf{false}$ and go to the next step, else continue decrypting.
2. If $\mathsf{flg}_{\mathtt{valid}} = \mathsf{true}$, post $(\mathtt{valid}, \mathsf{Sort}(\langle \mathtt{M}_j\rangle_{j\in[n]}))$ on the BB.
3. If $\mathsf{flg}_{\mathtt{valid}} = \mathsf{false}$, let $\mathcal{L}_0 = \langle \mathtt{C}_j^*\rangle_{j\in[n]}$ be the list of ciphertexts posted by the senders. For $j = 1$ to n, decrypt $\mathtt{M}_j \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathtt{C}_j^*)$ and post $(\mathtt{invalid}, \langle \mathtt{M}_j\rangle_{j\in[n]})$ on the BB.

Audit Message. The mixers $\mathcal{P}_{M_i}$ agree on posting the message $(\mathsf{flg}_{\mathtt{valid}})$ on the BB.

Algorithm Audit: This simply consists into reading from the BB and computing the Verification step of the protocol above (notice that this only relies on public information). The algorithm outputs 1 if it recomputes the same flag $\mathsf{flg}_{\mathtt{valid}}$ posted at the end of the BB.

5.2 Description of the Algorithms

We describe a protocol that makes use of the ideal functionality $\mathcal{F}^{\mathsf{Dec}}$ and $\mathcal{F}^{\mathsf{NPRO}}$. We implicitly make the parties call the functionality $\mathcal{F}^{\mathsf{NPRO}}$ when they execute the algorithms $\mathsf{Enc}, \mathsf{ADec}$. We write $(\mathtt{read}, (i, j))$ as a shorthand for the set of messages $\{(\mathtt{read}, k)\}_{k\in\{i,\ldots,j\}}$.

[5] In fact, the resource $\mathcal{F}^{\mathtt{Mix}}$ for n senders can be used to realize the functionality $\mathcal{F}^{\mathtt{Mix}}$ for $n' \leq n$ senders where the value n' is an influence of the adversary.

Protocol Π:

1. Input Submission. For all $j \in [n]$ the sender $\mathcal{P}_{S_j}$ upon input $\mathtt{M}_j$:
   (a) Send the message $\mathtt{pk}$ to the port $\mathtt{Dec.in}_{S_j}$ and return the activation;
   (b) (At next activation) read $\mathtt{pub} = (\mathsf{pk}, \mathsf{crs}_{\mathtt{sd}}, \mathsf{crs}_{\mathtt{mx}})$ from the port $\mathtt{Dec.out}_{S_j}$;
   (c) Compute $\mathtt{C}_j \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathtt{M}_j; R_j, w_j, x_j, y_j)$ by sampling randomness R_j, w_j, x_j, y_j for the encryption procedure (as described in Sec. 4.3);
   (d) Parse $\mathtt{C}_j = (\bar{W}_j, \bar{X}_j, \bar{Y}_j)$ and compute $\Pi_j^{\mathtt{s}} \leftarrow\!\!\$\ \mathsf{P}_{\mathtt{sd}}(\mathsf{crs}_{\mathtt{sd}}, (j, [\mathbf{g}, a], \bar{W}_j), (w_j, R_j))$;
   (e) Send $(\mathtt{write},\ (\mathtt{input}, j, (\mathtt{C}_j, \Pi_j^{\mathtt{s}}))\ )$ to the outport $\mathtt{BB.in}_{S_j}$ and return.

2. Optimistic Mix. For all $i = 1$ to m, in order, the mixer $\mathcal{P}_{M_i}$ does the following:
   (a) Sample a permutation $\pi_i \leftarrow\!\!\$\ \mathcal{S}_n$, and a secret randomization key $\kappa_i \leftarrow\!\!\$\ \mathbb{Z}_q$, and send $(\mathtt{query}, \kappa_i)$ to the outport $\mathtt{NPRO.in}_{M_i}$;
   (b) (At next activation) read com_i from the inport $\mathtt{NPRO.out}_{M_i}$;
   (c) If this is the first mixer, i.e., $i = 1$, then send $(\mathtt{read}, (1, n))$ to the outport $\mathtt{BB.in}_{M_i}$ and return;
   (d) (At next activation) if $i = 1$ read the messages $\{(\mathtt{C}_j, \Pi_j^{\mathtt{s}})\}_{j\in[n]}$ from the inport $\mathtt{BB.out}_{M_i}$, set $\mathcal{L}_0 \leftarrow \langle \mathtt{C}_{0,j} \rangle_{j\in[n]}$, and send[6] $(\mathtt{write}, \mathcal{L}_0)$ to $\mathtt{BB.in}_{M_0}$;
   (e) If $i > 1$ then send $(\mathtt{read}, n + i - 1)$ to the outport $\mathtt{BB.in}_{M_i}$ and return;
   (f) If $i > 1$ read $\mathcal{L}_{i-1}$ from the inport of $\mathtt{BB}$ and parse it as $\langle \mathtt{C}_{i-1,j} \rangle_{j\in[n]}$;
   (g) For all $j \in [n]$, sample $w_j, y_j, s_j, t_j \leftarrow\!\!\$\ \mathbb{Z}_q$ at random, compute:

$$\mathtt{C}_{i,\pi_i(j)} \leftarrow \mathsf{ARand}(\mathsf{pk}, \kappa_i, \mathtt{C}_{i-1,j}; w_j, y_j, s_j, t_j);$$

   and set $\mathcal{L}_i \leftarrow \langle \mathtt{C}_{i,j} \rangle_{j\in[n]}$;
   (h) Send the message $(\mathtt{write}, (\mathcal{L}_i, \mathsf{com}_i))$ to the outport $\mathtt{BB.in}_{M_i}$;
   (i) Compute the value $\hat{w}_i = \sum_{j=1}^{n} w_j$, store it, and return.
3. Prove. After all mixers are done with the previous phase, every mixer $\mathcal{P}_{M_i}$ does the following in parallel:
   (a) Compute the proof $\Pi_i^{\mathtt{M}} \leftarrow\!\!\$\ \mathsf{P}_{\mathtt{mx}}(\mathsf{crs}_{\mathtt{mx}}, (i, [\mathbf{g}, a], \mathbf{CkSum}(\mathcal{L}_{i-1}, \mathcal{L}_i, \kappa_i)), \hat{w}_i)$;
   (b) Send the message $(\mathtt{write}, (\Pi_i^{\mathtt{M}}, \kappa_i))$ to the outport $\mathtt{BB.in}_{M_i}$ and return;
4. Verification. Every mixer $\mathcal{P}_{M_i}$ executes the following in parallel:
   (a) Send the message $(\mathtt{read}, (n + 1, n + 2m + 1))$ to the outport $\mathtt{BB.in}_{M_i}$ and return;
   (b) (At next activation) read $\mathcal{L}_0, \{(\mathcal{L}_i, \mathsf{com}_i)\}_{i\in[m]}, \{\Pi_i^{\mathtt{M}}, \kappa_i\}_{i\in[m]}$ from inport $\mathtt{BB.out}_{M_i}$;
   (c) If $\exists j \in [m]$ s.t. $\mathtt{RO}(\kappa_j) \neq \mathsf{com}_j$ or $\mathsf{V}_{\mathtt{mx}}(\mathsf{crs}_{\mathtt{mx}}, (j, [\mathbf{g}, a], \mathbf{CkSum}(\mathcal{L}_{j-1}, \mathcal{L}_j, \kappa_j)), \Pi_j^{\mathtt{M}}) = 0$, then set the flag $\mathsf{flg}_{\mathtt{valid}}$ to false;
   (d) Send the message $(\mathtt{read}, 1, n)$ to the outport $\mathtt{BB.in}_{M_i}$ and return;
   (e) (At next activation) read the messages $\{\mathtt{C}_j, \Pi_j^{\mathtt{s}}\}_{j\in[n]}$ from the inport $\mathtt{BB.out}_{M_i}$;
   (f) If $\exists j \in [n]$ s.t. $\mathsf{V}_{\mathtt{sd}}(\mathsf{crs}_{\mathtt{sd}}, (j, [\mathbf{g}, a], \bar{W}_j), \Pi_j^{\mathtt{s}}) = 0$ then set the flag $\mathsf{flg}_{\mathtt{valid}}$ to false (where $\bar{W}_j$ is the first component of ciphertext $\mathtt{C}_j$);
   (g) Compute $\kappa^* \leftarrow \prod_{j=1}^{m} \kappa_j$;
5. Decode. All the mixer parties $\mathcal{P}_{M_i}$ executes the following in parallel:
   (a) if the flag $\mathsf{flg}_{\mathtt{valid}}$ is true send $(\mathtt{read}, n+m+1)$ to the outport $\mathtt{BB.in}_{M_i}$ and return; else send $(\mathtt{read}, n + 1)$ to the outport $\mathtt{BB.in}_{M_i}$ and return;
   (b) (At next activation) if $\mathsf{flg}_{\mathtt{valid}}$ is true read the message $(\mathcal{L}_m, \mathsf{com}_m)$ from $\mathtt{BB.out}_{M_i}$ and parse $\mathcal{L}_m = \langle \mathtt{C}_j^* \rangle_{j\in[n]}$, else read the message $\mathcal{L}_0$, parse $\mathcal{L}_0 = \langle \mathtt{C}_j^* \rangle_{j\in[n]}$, and set $\kappa^* \leftarrow 1$.
   (c) For all $j \in [n]$ send the message $(\mathtt{dec}, \kappa^*, \mathtt{C}_j^*)$ to the outport $\mathtt{Dec.in}_{M_i}$; return the activation;

[6] This message is redundant and not necessary for the security of the scheme, however we add it to simplify the exposition of the scheme.

(d) (At next activation) read the messages $\langle \mathtt{M}_j \rangle_{j\in[n]}$ from the outport $\mathtt{Dec.out}_{M_i}$; If $\mathsf{flg}_{\mathtt{valid}}$ is true and there exists j such that $\mathtt{M}_j = \perp$ then set the flag $\mathsf{flg}_{\mathtt{valid}}$ to false and go back to step 5 (restart the decoding); else if $\mathsf{flg}_{\mathtt{valid}}$ is true and $\mathtt{M}_j \neq \perp$ $\forall j \in [n]$, then send $(\mathtt{write}, \mathtt{valid}, \mathsf{Sort}(\langle \mathtt{M}_j \rangle_{j\in[n]}))$ to $\mathtt{BB.in}_{M_i}$ and return;

(e) if $\mathsf{flg}_{\mathtt{valid}}$ is false send the message $(\mathtt{write}, \mathtt{invalid}, \langle \mathtt{M}_j \rangle_{j\in[n]})$.

6. Audit Message. The mixers $\mathcal{P}_{M_i}$ agree on sending one message $(\mathtt{write}, \kappa^*)$ and one message $(\mathtt{write}, \mathsf{flg}_{\mathtt{valid}})$.

Algorithm Audit: On input the transcripts $\mathcal{D}^R = \{\mathsf{pub}, \{X_j\}_{j\in[n]}, \{\mathcal{L}_i, \mathsf{com}_i\}_{i\in[m]}, \{\Pi_i^{\mathtt{M}}, \kappa_i\}_{i\in[m]}, \kappa^*, \mathsf{flg}_{\mathtt{valid}}\}$, $\mathcal{D}^I$ and $(b, \mathcal{O})$. Output 1 if and only if $\mathsf{flg}_{\mathtt{valid}} = b$ and either $b = \mathtt{valid}$ all all the conditions below hold or $b = \mathtt{invalid}$ and at least one of the conditions below is false.

1. For all $j \in [n]$ parse $X_j = (\mathtt{C}_j, \Pi_j^{\mathtt{s}})$ and check if $\mathsf{V}_{\mathtt{sd}}(\mathsf{crs}_{\mathtt{sd}}, (j, [\mathbf{g}, a], \bar{W}_j), \Pi_j^{\mathtt{s}})$ is true where $\bar{W}_j$ is the first component of ciphertext $\mathtt{C}_j$.
2. For all $j \in [m]$ check if $\mathsf{V}_{\mathtt{mx}}(\mathsf{crs}_{\mathtt{mx}}, (j, [\mathbf{g}, a], \mathbf{CkSum}(\mathcal{L}_j, \mathcal{L}_{j+1}, v_j)), \Pi_j^{\mathtt{M}})$ is true.
3. If $b^* = \mathtt{valid}$, parse $\mathcal{L}_m = \langle \mathtt{C}_j^* \rangle_{j\in[n]}$ else $\mathcal{L}_0 = \langle \mathtt{C}_j^* \rangle_{j\in[n]}$ and for all $j \in [n]$ check that there exists $\mathtt{M}_j^*$ such that $\mathsf{Audit}^*_{\mathtt{Dec}}(\mathcal{D}^I, (\mathtt{dec}, \mathtt{C}_j^*, \kappa^*, \mathtt{M}_j^*))$ and check that $\mathsf{Sort}(\langle \mathtt{M}_j^* \rangle_{j\in[n]}) = \mathcal{O}$.

Theorem 6. *The protocol* (Π, Audit)*described above is* $(\mathcal{F}^{\mathsf{Dec}}, \mathcal{F}^{\mathsf{NPRO}})$*-secure OMix.*

Proof. The first part of the proof consists in showing a simulator S and arguing that for any PPT environment $\mathcal{Z}$ it holds

$$(\Pi \circ \mathcal{F}^{\mathsf{Dec}} \circ \mathcal{F}^{\mathsf{NPRO}} \circ \mathrm{BB} \circ \mathcal{Z}) \approx_c (\mathcal{F}^{\mathtt{OMix}} \circ \mathsf{S} \circ \mathcal{Z}).$$

To show the indistinguishability of the above experiments we give a sequence of hybrid experiments in which the experiment running the interactive agent $(\Pi \circ \mathcal{F}^{\mathsf{Dec}} \circ \mathcal{F}^{\mathsf{NPRO}} \circ \mathrm{BB} \circ \mathcal{Z})$ is progressively modified until reaching an experiment that is identically distributed to the one with our simulator S.

In the proof, we let h^* be the index of the first honest mixer, i.e., h^* is the smallest index in $[m] \setminus \mathbf{C}^M$. Also, our simulator and hybrid experiments maintain two sets Ψ_{in} and Ψ_{hide}, both consisting of tuples $(X, Y) \in (\mathbb{G}^\ell \cup \{\diamond\})^2$. For Ψ_{in} (resp. Ψ_{hide}) we define a corresponding *map* $\psi_{\mathsf{in}} : \mathbb{G}^\ell \to \mathbb{G}^\ell \cup \{\diamond, \perp\}$ (resp. ψ_{hide}) such that

$$\psi_{\mathsf{in}}(X) = \begin{cases} Y & \text{if } (X, Y) \in \Psi_{\mathsf{in}} \\ X & \text{else} \end{cases}$$

and analogously for ψ_{hide}. Moreover, both the simulator and the hybrid experiments keep track of all the queries to the random oracle $\mathcal{F}$ in a set $\mathcal{Q}^{\mathcal{F}}$ of pairs $(\kappa, \mathsf{com}) \in \mathbb{Z}_q \times \{0,1\}^\lambda$.

We proceed with a sequence of hybrids. Before describing them in detail, we review them informally:

Hybrid $\mathbf{H}_0$: We check that for every value com_i posted by malicious mixers on the BB there exists a pre-image κ_i in the list of the queries to the random oracle $\mathtt{RO}$, otherwise we abort. By the random oracle property, this hybrid is statistically close to the real experiment.

Hybrid $\mathbf{H}_1$: We generate the CRS of the NIZKs using the trapdoor mode. By the zero-knowledge property this hybrid is computationally indistinguishable from the previous one.

Hybrid $\mathbf{H}_2$: We introduce the set $\hat{\mathcal{M}}_H$ of honest simulated messages (each randomly chosen) and two empty lists $\Psi_{\mathsf{in}}, \Psi_{\mathsf{hide}}$. The list Ψ_{in} is populated to map each simulated honest input in $\hat{\mathcal{M}}_H$ to a corresponding real honest input, and the decryption agent is modified to output $\psi_{\mathsf{in}}(\psi_{\mathsf{hide}}(\mathtt{M}))$ instead of $\mathtt{M}$. This hybrid is distributed the same as the previous one except if at

decryption time a message in $\hat{\mathcal{M}}_H$ is hit (in this case the map ψ_{in} would modify the returned value). However, since messages in $\hat{\mathcal{M}}_H$ are randomly chosen and are not in the environment's view, this bad event happens only with negligible probability.

Hybrid $\mathbf{H}_3$: We encrypt the simulated sender inputs $\hat{\mathtt{M}}_j$ instead of the the honest sender inputs. This hybrid can be shown indistinguishable from the previous one based on the RCCA security of the PKE scheme, and the zero-knowledge of $\mathcal{NIZK}_{\mathtt{sd}}$. The goal of the changes done so far is that from now on we can keep track of the every honest ciphertext via its underlying message $\hat{\mathtt{M}}_j \in \hat{\mathcal{M}}_H$, which acts as a unique handle for it. As we shall see, this for example allows us to understand if a mixer "removed" a ciphertext by checking if the corresponding handle has disappeared when looking at the decryptions of the ciphertexts returned by that mixer.

Hybrid $\mathbf{H}_4$: Let $\mathcal{V}_m$ be the decryption of the list of ciphertexts output by the last mixer $\mathcal{P}_{M_m}$. If the protocol execution is valid and the set $\hat{\mathcal{M}}_H$ is not a subset of $\mathcal{V}_m$, $\mathbf{H}_4$ aborts. This check essentially ensures that none of the inputs of the honest senders has been discarded. Using the adaptive soundness of $\mathcal{NIZK}_{\mathtt{mx}}$, the zero-knowledge and simulation extractability of $\mathcal{NIZK}_{\mathtt{sd}}$, and the RCCA security of our PKE, we can show that this abort can happen only with negligible probability. Notice that if there is no abort we are guaranteed that $\mathcal{V}_m$ contains all the messages of the honest senders, i.e., $\hat{\mathcal{M}}_H \subseteq \mathcal{V}_m$.

Hybrid $\mathbf{H}_5$: This hybrid checks if in $\mathcal{V}_m$ every message of $\hat{\mathcal{M}}_H$ appears only once, and aborts if not. Intuitively, this check ensures that none of the inputs of the honest sender was copied. Similarly to the previous hybrid, we show how to reduce an adversary causing such an abort again to adaptive soundness of $\mathcal{NIZK}_{\mathtt{mx}}$, the zero-knowledge and simulation extractability of $\mathcal{NIZK}_{\mathtt{sd}}$, and the RCCA security of the PKE.

Hybrid $\mathbf{H}_6$: Let $\mathcal{V}_{h^*}$ be decryption of the list of ciphertexts output by the first honest mixer. If the protocol execution is valid and $\mathcal{V}_m \not\subseteq \mathcal{V}_{h^*}$, $\mathbf{H}_6$ aborts. Intuitively, this check ensures that the set of ciphertexts output by the last mixer is a subset of the set of ciphertexts returned and *authenticated* by the first honest mixers. In a nutshell, this means for example that an adversary controlling the mixers after $\mathcal{P}_{M_{h^*}}$ the best it can do with ciphertexts corresponding to honest senders is to invalidate them. We show that an adversary causing such an abort can be reduced to attack against the UF-CCA property of our PKE scheme.
Notice that with the changes done so far, if no abort occurs, we are ensured that $\hat{\mathcal{M}}_H \subseteq \mathcal{V}_m \subseteq \mathcal{V}_{h^*}$, which also means that the list of ciphertexts returned by the last mixer decrypts to a permutation of the messages of the honest senders.
The next hybrids are aimed to show that this permutation is hidden to the environment.

Hybrid $\mathbf{H}_7$: We simulate the zero-knowledge proof of the first honest mixer. By the zero-knowledge of $\mathcal{NIZK}_{\mathtt{mx}}$, this hybrid is indistinguishable from the previous one.

Hybrid $\mathbf{H}_8$: Let $\langle \mathtt{C}_{h^*-1,j} \rangle_{j\in[n]}$ be the list of ciphertexts received by the first honest mixer $\mathcal{P}_{M_{h^*}}$. Instead of re-randomizing all ciphertexts, here $\mathcal{P}_{M_{h^*}}$ decrypts and re-encrypts all the valid ones, i.e., it sets $\mathtt{C}_{h^*,\pi_{h^*}(j)}$ as a fresh encryption of $\mathtt{M}_{h^*-1,j}$, if $\mathtt{M}_{h^*-1,j} \neq \perp$. This difference can be reduced to the (weak) RCCA re-randomizability of our PKE scheme. This hybrid is preparatory for the next hybrid.

Hybrid $\mathbf{H}_9$: Here, instead of re-encrypting the same messages, we re-encrypt new fresh (and uncorrelated) messages. Namely, instead of creating $\mathtt{C}_{h^*,\pi_{h^*}(j)}$ as a re-encryption of $\mathtt{M}_{h^*-1,j}$, this is set as an encryption of a random an independent message $\mathtt{H}_j$. Moreover, we start populating the set Ψ_{hide} to associate $\mathtt{H}_j$ with $\mathtt{M}_{h^*-1,j}$; this way this modification is not visible by looking at the outcome of decryption or because of the changes introduced in $\mathbf{H}_4, \mathbf{H}_5, \mathbf{H}_6$. The indistinguishability of $\mathbf{H}_8$ and $\mathbf{H}_9$ can be reduced to the RCCA security of the PKE.

Hybrid $\mathbf{H}_{10}$: We do not permute anymore according to π_{h^*} the list of ciphertexts returned by the first honest mixer, i.e., $\mathtt{C}_{h^*,j}$ is a re-encryption of $\mathtt{H}_j$. Precisely, this is done only for those indices j such that $\mathtt{C}_{h^*-1,\pi^{-1}_{h^*}(j)}$ decrypts correctly. We show that this hybrid is identically

distributed to the previous one; intuitively because we are just permuting random messages that, until that point, were not in the view of the environment. This makes the list of ciphertexts returned by $\mathcal{P}_{M_{h^*}}$ completely independent from the permutation π_{h^*} restricted to the indices for which $\mathtt{C}_{h^*,j}$ decrypts correctly, which include all the indices of the honest senders.

In what follows we describe each hybrid experiment in detail, and prove the indistinguishability of each consecutive pair.

Hybrid 0. Let the hybrid experiment $\mathbf{H}_0$ be the experiment that runs the interactive agent $(\Pi \circ \mathcal{F}^{\mathsf{Dec}} \circ \mathcal{F}^{\mathsf{NPRO}} \circ \mathrm{BB} \circ \mathcal{Z})$. More precisely, the hybrid sees the internal views of all the agents, and additionally keeps track of all the messages sent by $\mathcal{Z}$ to the random oracle functionality $\mathcal{F}^{\mathsf{NPRO}}$. Let $\mathcal{Q}^{\mathcal{F}}$ be the set of queries made by $\mathcal{Z}$ to $\mathcal{F}$. $\mathbf{H}_0$ does the following operations:

1. Initialize the flag $\mathsf{flg}_{\mathsf{RO}}$ to false;
2. For any $i \in \mathbf{C}^M$ when the message $(\mathtt{write}, (\mathcal{L}_i, \mathsf{com}_i))$ appears in the port $\mathtt{BB.in}_{M_i}$ check if there exists an entry $(\tilde{\kappa}_i, \mathsf{com}_i)$ in the set $\mathcal{Q}^{\mathcal{F}}$; If the check fails then set $\mathsf{flg}_{\mathsf{RO}}$ to true;
3. If the flag $\mathsf{flg}_{\mathsf{RO}}$ true and the message $(\mathtt{write}, \mathtt{valid})$ appears in a port $\mathtt{BB.in}_{M_i}$ where $i \in [n] \setminus \mathbf{C}^M$ then **abort**.

Lemma 35. $\mathbf{H}_0 \approx_s (\Pi \circ \mathcal{F}^{\mathsf{Dec}} \circ \mathcal{F}^{\mathsf{NPRO}} \circ \mathrm{BB} \circ \mathcal{Z})$.

Proof. The two experiments diverge if the event in the step 3 above happens. Let us call `GuessRO` such event. By Lemma 1, we need to prove that $\Pr[\mathtt{GuessRO}] \in \mathtt{negl}(\lambda)$. To see this, notice that if `GuessRO` occurs, then during the verification phase (Step 4 of Π) the message $(\mathtt{write}, (\Pi_i^{\mathtt{M}}, \kappa_i))$ appears in $\mathtt{OMix.infl}_{M_i}$ and $\mathcal{F}(\kappa_i) = \mathsf{com}_i$. This means that the environment $\mathcal{Z}$ guessed the image of κ_i for $\mathcal{F}$ before $\mathcal{F}$ was queried on this value. The latter event happens only with probability $2^{-\lambda}$.

Hybrid 1. The hybrid experiment $\mathbf{H}_1$ is the same as $\mathbf{H}_0$ but with the following difference:

1. The agent of the ideal functionality $\mathcal{F}^{\mathsf{Dec}}$, instead of steps 2 and 3, computes $\mathsf{crs}_{\mathtt{sd}}, \mathsf{td}^s_{\mathtt{sd}}, \mathsf{td}^e_{\mathtt{sd}} \leftarrow \overline{\mathsf{Init}_{\mathtt{sd}}}(1^\lambda)$ and $\mathsf{crs}_{\mathtt{mx}}, \mathsf{td}^s_{\mathtt{mx}} \leftarrow \overline{\mathsf{Init}_{\mathtt{mx}}}(1^\lambda)$ respectively;

Lemma 36. $\mathbf{H}_1 \approx_c \mathbf{H}_0$.

The indistinguishability can be reduced in a straightforward way to the zero-knowledge of $\mathcal{NIZK}_{\mathtt{sd}}$ and $\mathcal{NIZK}_{\mathtt{mx}}$.

Hybrid 2. The hybrid experiment $\mathbf{H}_2$ runs the same of $\mathbf{H}_1$ but with the following difference:

1. Compute the list of inputs of the honest senders $\mathcal{M}_H \leftarrow \{\mathtt{M}_j : j \in [n] \setminus \mathbf{C}^S\}$ (this can be easily computed by watching at the port $\mathtt{Omix.in}_{S_j}$ for $j \in [n] \setminus \mathbf{C}^S$) and sample the list of simulated inputs of the honest senders $\hat{\mathcal{M}}_H \leftarrow \{\hat{\mathtt{M}}_j : j \in [n] \setminus \mathbf{C}^S\}$ such that each $\hat{\mathtt{M}}_j \leftarrow_{\$} \mathbb{G}^\ell$ is sampled uniformly at random and $\hat{\mathcal{M}}_H$ contains no repetitions;
   For every $j \in [n] \setminus \mathbf{C}^S$, add $(\hat{\mathtt{M}}_j, \mathtt{M}_j)$ to Ψ_{in}, and initialize Ψ_{hide} as the empty set.
2. Whenever a message $(\mathtt{dec}, X, \kappa, \mathtt{M})$ appears either in the leakage port `Dec.lk` or in the outport $\mathtt{Dec.out}_{M_i}$ for any i, replace the message with $(\mathtt{dec}, X, \kappa, \psi_{\mathsf{in}}(\psi_{\mathsf{hide}}(\mathtt{M})))$;

Lemma 37. $\mathbf{H}_2 \approx_s \mathbf{H}_1$.

Proof. Notice that $\mathbf{H}_2$ differs from $\mathbf{H}_1$ only when $\mathtt{M}$ is replaced with $\psi_{\mathsf{in}}(\psi_{\mathsf{hide}}(\mathtt{M}))$ in step (2). However, this replacement causes a change only if $\mathtt{M} \in \hat{\mathcal{M}}_H$. We argue that the probability of this event is upper bounded by $n^2/(|\mathbb{G}|^\ell - n) \in \mathtt{negl}(\lambda)$. To see this, observe that prior to the occurrence of this event, the view of $\mathcal{Z}$ in $\mathbf{H}_2$ is independent from the values in $\hat{\mathcal{M}}_H$, and thus the adversary has n trials to guess one of the possible n elements chosen at random from a set of size $|\mathbb{G}|^\ell$ and for each (unsuccessful) trial it can exclude one possible value.

Hybrid 3. The hybrid experiment $\mathbf{H}_3$ runs the same of $\mathbf{H}_2$ but with the following difference:

1. Whenever a message $(\texttt{input}, j, (\mathtt{C}_j, \Pi_j^{\mathtt{s}}))$ appears in a port $\mathtt{BB.in}_{S_j}$ for $j \in [n] \setminus \mathbf{C}^S$ then replace the message with $(\texttt{input}, j, (\hat{\mathtt{C}}_j, \hat{\Pi}_j^{\mathtt{s}}))$ where $\hat{\mathtt{C}}_j \leftarrow \mathsf{Enc}(\mathsf{pk}, \hat{\mathtt{M}}_j; R_j, w_j, x_j, y_j, s_j, t_j)$ and $\hat{\Pi}_j^{\mathtt{s}} \leftarrow\!\!\$\ \mathsf{P}_{\mathtt{sd}}(\mathsf{crs}_{\mathtt{sd}}, (j, [\mathbf{g}, a], \bar{W}_j), (w_j, Rj))$, $R_j \leftarrow\!\!\$\ \mathbb{G}$, $w_j, x_j, y_j, s_j, t_j \leftarrow\!\!\$\ \mathbb{Z}_q$ and $\bar{W}_j \leftarrow (w_j \cdot [\mathbf{g}], w_j \cdot [a] + R_j)$;

Lemma 38. $\mathbf{H}_3 \approx_c \mathbf{H}_2$.

Proof. The indistinguishability of the two hybrids can be argued based on the RCCA-security of $\mathcal{PKE}$ and zero-knowledge of $\mathcal{NIZK}_{\mathtt{sd}}$ via a sequence of hybrid experiments in which the encrypted honest messages are replaced one by one. This reduction is rather standard, so we provide here only a sketch.

For $i \in [n]$, let $\mathbf{H}_{3,i}$ be the hybrid that executes the same code of $\mathbf{H}_3$ for all the honest senders $\mathcal{P}_{S_j}$ such that $j < i$, and executes the code prescribed by the protocol for all the honest senders $\mathcal{P}_{S_j}$ such that $j \geq i$. Clearly, $\mathbf{H}_{3,1} \equiv \mathbf{H}_2$ and $\mathbf{H}_{3,n} \equiv \mathbf{H}_3$. Now we only need to show that $\mathbf{H}_{3,i} \approx_c \mathbf{H}_{3,i+1}$. To do so we first define another hybrid $\mathbf{H}'_{3,i}$ that executes the same code of $\mathbf{H}_{3,i}$ except that the proof $\Pi_i^{\mathtt{s}}$ is computed using the simulator of $\mathcal{NIZK}_{\mathtt{sd}}$ and the knowledge of $\mathsf{td}^s_{\mathtt{sd}}$. Clearly, $\mathbf{H}_{3,i} \approx_c \mathbf{H}'_{3,i}$ holds based on the adaptive zero-knowledge property of $\mathcal{NIZK}_{\mathtt{sd}}$. Similarly, let $\mathbf{H}''_{3,i+1}$ be the same as $\mathbf{H}_{3,i+1}$ except that the proof $\Pi_i^{\mathtt{s}}$ is computed using the simulator of $\mathcal{NIZK}_{\mathtt{sd}}$. Then we are left with proving $\mathbf{H}'_{3,i} \approx_c \mathbf{H}''_{3,i+1}$, and this we reduce it to RCCA-security.

Briefly, we can define an adversary $\mathcal{B}$ that runs $\mathbf{H}'_{3,i}$ and instead of running the code of $\mathcal{P}_{S_i}$ asks the challenger with challenge plaintexts $(\mathtt{M}_i, \hat{\mathtt{M}}_i)$. Notice that at this point the proof $\Pi_i^{\mathtt{s}}$ can be computed without knowing the randomness of the ciphertext, as it is simulated. Next, $\mathcal{B}$ continues the execution of $\mathbf{H}'_{3,i}$ but uses the decryption oracle whenever a decryption is necessary. Notice that, thanks to the modification introduced in $\mathbf{H}_2$, $\mathcal{B}$ can safely return $\mathtt{M}_i$ whenever the decryption oracle returns $\diamond$. Indeed, by the definition of the map ψ_{in} we have $\psi_{\mathsf{in}}(\hat{\mathtt{M}}_i) = \mathtt{M}_i = \psi_{\mathsf{in}}(\mathtt{M}_i)$. $\mathcal{B}$'s simulation is perfect: if $\mathtt{M}_i$ (resp. $\hat{\mathtt{M}}_i$) is chosen by its challenger, $\mathcal{B}$ simulates $\mathbf{H}'_{3,i}$ (resp. $\mathbf{H}''_{3,i+1}$).

Hybrid 4. The hybrid experiment $\mathbf{H}_4$ is as the hybrid $\mathbf{H}_3$ but with the following difference:

1. At initialization phase, $\mathsf{flg}_{\mathtt{stuffed}}$ is initialized to false;
2. Executing the step 2 of the protocol (Optimistic Mix), recall that for any $i \in [m]$, $\mathcal{L}_i = \langle \mathtt{C}_{i,1}, \ldots, \mathtt{C}_{i,n} \rangle$ is the list of ciphertext produced by the mixer $\mathcal{P}_{M_i}$ and $\mathcal{L}_0 = \langle \mathtt{C}_{0,1}, \ldots, \mathtt{C}_{0,n} \rangle$ consists of all the ciphertexts submitted by the senders.
   Compute the multiset $\mathcal{V}_0 \leftarrow \{\mathsf{Dec}(\mathsf{sk}, \mathtt{C}_{0,j}) : j \in [n]\}$.
3. When the message $(\texttt{write}, (\mathcal{L}_m, \mathsf{com}_m))$ appears in the port $\mathtt{BB.in}_{M_m}$, compute the multiset $\mathcal{V}_m \leftarrow \{\mathsf{ADec}(\mathsf{sk}, \prod_{l=1}^m \kappa_l, \mathtt{C}_{m,j}) : j \in [n]\}$.
   If $\hat{\mathcal{M}}_H \not\subseteq \psi_{\mathsf{hide}}(\mathcal{V}_m)$ then set the flag $\mathsf{flg}_{\mathtt{stuffed}}$ to true.
4. At the end of the protocol's execution, if $\mathsf{flg}_{\mathtt{stuffed}}$ is true and a the message $(\texttt{write}, \texttt{valid})$ appears in a port $\mathtt{BB.in}_{M_i}$ where $i \in [n] \setminus \mathbf{C}^M$, then **abort**;

Lemma 39. $\mathbf{H}_4 \approx_c \mathbf{H}_3$.

Proof. We prove the indistinguishability based on the RCCA-Security of $\mathcal{PKE}$ (as well as the simulation extractability and zero-knowledge of $\mathcal{NIZK}_{\mathtt{sd}}$ and adaptive soundness of $\mathcal{NIZK}_{\mathtt{mx}}$). To do so we define a sequence of intermediate hybrid experiments. Let $\mathbf{H}_{4,i}$ be the same as $\mathbf{H}_4$ but where the step 3 is substituted by the following:

3. If $\underline{\{\hat{\mathtt{M}}_j : j \in [i] \setminus \mathbf{C}^S\} \not\subseteq \psi_{\mathsf{hide}}(\mathcal{V}_m)}$ then set the flag $\mathsf{flg}_{\mathtt{stuffed}}$ to true.

Notice that $\mathbf{H}_{4,0} \equiv \mathbf{H}_3$ as the flag $\mathsf{flg}_{\mathtt{stuffed}}$ is never raised ($\emptyset \not\subseteq \mathcal{V}_m$ is always false), while $\mathbf{H}_{4,n} \equiv \mathbf{H}_4$.

Now, let $\mathbf{H}^{s_i}_{4,i-1}$ be the same as $\mathbf{H}_{4,i-1}$ but where the proof $\Pi^{\mathtt{s}}_i$ is computed using the simulator of $\mathcal{NIZK}_{\mathtt{sd}}$ and the knowledge of $\mathsf{td}^s_{\mathtt{sd}}$ and let $\mathbf{H}^{s_i}_{4,i}$ be the same as $\mathbf{H}_{4,i}$ but where the proof $\Pi^{\mathtt{s}}_i$ is computed using the simulator of $\mathcal{NIZK}_{\mathtt{sd}}$ and the knowledge of $\mathsf{td}^s_{\mathtt{sd}}$. By zero-knowledge property of $\mathcal{NIZK}_{\mathtt{sd}}$ we have $\mathbf{H}_{4,i-1} \approx_c \mathbf{H}^{s_i}_{4,i-1}$ and $\mathbf{H}^{s_i}_{4,i} \approx_c \mathbf{H}_{4,i}$.

Next, let $\mathbf{H}^{s_i,e}_{4,i-1}$ (resp. $\mathbf{H}^{s_i,e}_{4,i}$) be the same as $\mathbf{H}^{s_i}_{4,i-1}$ (resp. $\mathbf{H}^{s_i}_{4,i}$) with the following additional step: whenever a message $(\mathtt{input}, j, (\mathtt{C}_j, \Pi^{\mathtt{s}}_j))$ appears in a port $\mathtt{Mix.infl}_{S_j}$, with $j \in \mathbf{C}^S$ (i.e., an input from the corrupted sender $\mathcal{P}_{S_j}$), if $\mathsf{V}_{\mathtt{sd}}(\mathsf{crs}_{\mathtt{sd}}, (j, [\mathbf{g}, a], \bar{W}_j), \Pi^{\mathtt{s}}_j) = 1$ compute $\hat{R}_j \leftarrow \mathsf{Ext}_{\mathtt{sd}}(\mathsf{td}^e_{\mathtt{sd}}, (j, [\mathbf{g}, a], \bar{W}_j), \Pi^{\mathtt{s}}_j)$, and if $\hat{R}_j = \perp$ then abort. By the simulation-extractability of $\mathcal{NIZK}_{\mathtt{sd}}$, we have $\mathbf{H}^{s_i}_{4,i-1} \approx_c \mathbf{H}^{s_i,e}_{4,i-1}$ and $\mathbf{H}^{s_i,e}_{4,i} \approx_c \mathbf{H}^{s_i}_{4,i}$.

Notice that we need to simulate only the proof $\Pi^{\mathtt{s}}_j$, thus we could base the indistinguishability of the hybrids steps just performed on the weaker notions of adaptive single-theorem zero-knowledge and one-time simulation extractability. For simplicity in the exposition however we use the stronger notions of multi-theorem zero-knowledge and simulation extractability.

We are left with showing $\mathbf{H}^{s_i,e}_{4,i-1} \approx_c \mathbf{H}^{s_i,e}_{4,i}$ $\forall i \in [n]$. The two hybrids differ when the optimistic mix succeeds and, for $i \notin \mathbf{C}^S$, $\hat{\mathtt{M}}_i \notin \mathcal{V}_m$ (notice that Ψ_{hide} is still empty). The first condition in particular implies that all the mixer proofs $\Pi^{\mathtt{M}}_j$ for $j \in \mathbf{C}^M$ and the senders proofs $\Pi^{\mathtt{s}}_j$ for $j \in \mathbf{C}^S$ verify and that all the decryptions performed by the protocol are valid (i.e., $\perp \notin \mathcal{V}_m$).

It is sufficient to bound the probability of the conjunction of the events:

1. $\mathtt{Verify}_0$: For all $j \in \mathbf{C}^M$, $\mathsf{V}_{\mathtt{mx}}(\mathsf{crs}_{\mathtt{mx}}, (j, [\mathbf{g}, a], \mathbf{CkSum}(\mathcal{L}_{j-1}, \mathcal{L}_j, \kappa_j)))$ is true,
2. $\mathtt{Verify}_1$: For all $j \in \mathbf{C}^S$, $\mathsf{V}_{\mathtt{sd}}(\mathsf{crs}_{\mathtt{sd}}, (j, [\mathbf{g}, a], \bar{W}_j))$ is true,
3. $\mathtt{Decrypt}$: $\perp \notin \mathcal{V}_m$,
4. $\mathtt{Stuffed}$: $\hat{\mathtt{M}}_i \notin \mathcal{V}_m$.

Let $\mathtt{Bad}$ be the conjunction of the events above. Consider the following adversary for the RCCA-Security experiment on $\mathcal{PKE}$:

Adversary $\underline{\mathsf{B}(\mathsf{pk})}$ with oracle access to $\mathsf{Dec}^{\diamond}$:

1. Run the hybrid $\mathbf{H}^{s_i,e}_{4,i}$ (but do not sample the parameter of the $\mathcal{PKE}$, instead use the value pk received as input);
2. Sample $\hat{\mathtt{M}}_{i,0}, \hat{\mathtt{M}}_{i,1}$ uniformly at random and send them to the challenger of the RCCA-experiment.
3. Receive the challenge ciphertext $\mathtt{C}^*$;
4. Once the agent $\mathcal{P}_{S_i}$ is activated send the message $(\mathtt{input}, (i, (\mathtt{C}^*, \Pi^*))$ where $\Pi^* \leftarrow \mathsf{Sim}_{\mathtt{sd}}(\mathsf{td}^s_{\mathtt{sd}}, \mathtt{C}^*)$ to the port of BB;
5. Compute all the decryptions using the oracle $\mathsf{Dec}^{\diamond}$; whenever the oracle answers $\diamond$, use the message $\mathtt{M}_i$ (the message provided by the honest sender, following in this way what the map ψ_{in} would do);
6. If either $\hat{\mathtt{M}}_i \in \mathcal{V}_{h^*}$ or the message $(\mathtt{write}, \mathtt{invalid})$ appears in a port $\mathtt{BB.in}_{M_j}$ where $j \in [n] \setminus \mathbf{C}^M$ then abort and output a random bit b;
7. Let $\mathcal{R} \leftarrow \{R_j : j \in [n] \setminus \{i\}\}$. (Recall that for $j \in [n] \setminus \{i\}$ if $j \in [n] \setminus \mathbf{C}^M$ then the value R_j is the sampled by the hybrid experiment else R_j is computed by the extractor of $\mathcal{NIZK}_{\mathtt{sd}}$.)
8. Let $(\mathtt{write}, \langle \mathtt{C}_{m,j} : j \in [n] \rangle)$ be the message appeared in the port $\mathtt{BB.in}_{M_m}$ and let $\mathcal{Q} \leftarrow \{R' : (R', *) \in \mathcal{Q}^{\mathcal{H}}\} \cup \mathcal{R}$, where $\mathcal{Q}^{\mathcal{H}}$ is the list of oracle queries made by $\mathcal{Z}$ to the random oracle $\mathcal{H}$.

For all $j \in [n]$, let $\tilde{\mathtt{M}}_j$ be the answer of the decryption oracle on $\tilde{\mathtt{C}}_j = \mathtt{C}_{m,j}/(\prod_{l=1}^{m} \kappa_l)$, and do the following:

For all $R' \in \mathcal{R}$:

- Compute $p = \mathcal{G}(\tilde{\mathtt{M}}_j | \mathcal{H}(R')))$ and $\bar{X}' = x \cdot [\mathbf{g}, \mathbf{b}, (pc + d)]$ for $x \leftarrow_{\$} \mathbb{Z}_q$.
- Query the decryption oracle on $\tilde{\mathtt{C}}_j + [\mathbf{0}, \bar{X}', \mathbf{0}]$.
- If the answer is not $\perp$ then set $\tilde{R}_j \leftarrow R'$ and break the cycle.

9. Compute $R^* \leftarrow (\sum_{j \in [n]} \tilde{R}_j) - (\sum_{j \in [n] \setminus \{i\}} R_j)$;
10. Compute $p \leftarrow \mathcal{G}(\hat{\mathtt{M}}_{i,0} | \mathcal{H}(R^*)))$ and $\bar{X}' \leftarrow x' \cdot [\mathbf{g}, \mathbf{b}, (pc + d)]$ for $x' \leftarrow_{\$} \mathbb{Z}_q$.
11. Query the decryption oracle on $\mathtt{C}^* + [\mathbf{0}, \bar{X}', \mathbf{0}]$.
12. If the answer is $\perp$ then output 1 else output 0.

We show that $\Pr\left[\mathbf{Exp}^{\mathtt{RCCA}}_{\mathsf{B},\mathcal{PKE}}(\lambda) = 1\right] \geq \frac{1}{2} + \frac{1}{2}\Pr\left[\mathtt{Bad}\right] - \mathtt{negl}(\lambda)$. Notice that

$$\Pr\left[\mathbf{Exp}^{\mathtt{RCCA}}_{\mathsf{B},\mathcal{PKE}}(\lambda) = 1\right] = \tfrac{1}{2}(1 - \Pr\left[\mathtt{Bad}\right]) + \Pr\left[\mathbf{Exp}^{\mathtt{RCCA}}_{\mathsf{B},\mathcal{PKE}}(\lambda) = 1 \wedge \mathtt{Bad}\right] \tag{2}$$

In fact, when $(\neg\mathtt{Bad})$ then in step 6 the adversary B outputs a random bit. Let $\mathtt{E}$ be the event "$\forall j \in \mathbf{C}^M : \mathbf{CkSum}(\mathcal{L}_{j-1}, \mathcal{L}_j, \kappa_j) \in \mathit{Span}([\mathbf{g}, a])$". Notice that:

$$\begin{aligned}\Pr[\mathbf{Exp}^{\mathtt{RCCA}}_{\mathsf{B},\mathcal{PKE}}(\lambda) = 1 \wedge \mathtt{Bad}] \geq \\ \Pr\left[\mathbf{Exp}^{\mathtt{RCCA}}_{\mathsf{B},\mathcal{PKE}}(\lambda) = 1 \mid \mathtt{Bad} \wedge \mathtt{E}\right] \cdot (\Pr\left[\mathtt{Bad}\right] - \Pr\left[\mathtt{Verify}_0 \wedge \neg\mathtt{E}\right]).\end{aligned}$$

Claim 11 $\Pr\left[\mathtt{Verify}_0 \wedge \neg\mathtt{E}\right] \leq \mathtt{negl}(\lambda)$.

Proof. For every $j \in \mathbf{C}^M$, let $\mathtt{E}_j$ be the event "$\mathbf{CkSum}(\mathcal{L}_{j-1}, \mathcal{L}_j, \kappa_j) \in \mathit{Span}([\mathbf{g}, a])$" so that $\mathtt{E} = \bigwedge_{j \in \mathbf{C}^M} \mathtt{E}_j$. Applying the union bound:

$$\Pr\left[\mathtt{Verify}_0 \wedge \neg\mathtt{E}\right] = \Pr\left[\bigvee_{j \in \mathbf{C}^M} (\mathtt{Verify}_0 \wedge \neg\mathtt{E}_j)\right] \leq \textstyle\sum_{j \in \mathbf{C}^M} \Pr\left[\mathtt{Verify}_0 \wedge \neg\mathtt{E}_j\right].$$

By the soundness of $\mathcal{NIZK}_{\mathtt{mx}}$, for any $j \in \mathbf{C}^M$ the probability of $\Pr\left[\neg\mathtt{Verify}_0 \wedge \neg\mathtt{E}_j\right]$ is negligible in λ.

Claim 12 $\Pr\left[\mathbf{Exp}^{\mathtt{RCCA}}_{\mathsf{B},\mathcal{PKE}}(\lambda) = 1 \mid \mathtt{Bad} \wedge \mathtt{E}\right] \geq 1 - \mathtt{negl}(\lambda)$.

Proof. We show that, conditioning on the equation above, the value $R^* = \sum_j \tilde{R}_j - \sum_{j \neq i} R_j$ is the randomizer of the challenge ciphertext with overwhelming probability. Since we condition on $\mathtt{E}$ we have $\mathbf{CkSum}(\mathcal{L}_0, \mathcal{L}_m, \prod_{l=1}^{m} \kappa_l) \in \mathit{Span}([\mathbf{g}, a])$. More in detail, for $k = 0, \ldots, m$, let $\mathcal{L}_k = \langle \mathtt{C}_{k,1}, \ldots, \mathtt{C}_{k,n} \rangle$ and recall that by construction of B we have $\mathtt{C}_{0,i} = \mathtt{C}^*$. If we let $\mathtt{C}_{i,j} = (\bar{W}_{i,j}, \bar{X}_{i,j}, \bar{Y}_{i,j})$, then there exists $t \in \mathbb{Z}_q$ such that:

$$\textstyle\sum_j \bar{W}_{m,j}/(\prod_{l=1}^{m} \kappa_l) + t \cdot [\mathbf{g}, a] = \sum_j \bar{W}_{0,j}$$

The equation above implies:

$$\textstyle\sum_j [-\mathbf{a}, 1]^T \cdot \bar{W}_{m,j}/(\prod_{l=1}^{m} \kappa_l) + \overbrace{[-\mathbf{a}, 1]^T \cdot t \cdot [\mathbf{g}, a]}^{=[0]} = \sum_j [-\mathbf{a}, 1]^T \cdot \bar{W}_{0,j} \tag{3}$$

First, notice that by the property of the $\mathcal{NIZK}_{\mathtt{sd}}$ extractor, by the modification added in $\mathbf{H}_3$, with overwhelming probability:

$$\sum_{j \neq i} R_j = \sum_{j \neq i} [-\mathbf{a}, 1]^T \cdot \bar{W}_{0,j}. \tag{4}$$

Second, notice that since $\perp \notin \mathcal{V}_m$ for any $\mathtt{C}_{m,j}$, with overwhelming probability, either there exists a value $(R', *) \in \mathcal{Q}^{\mathcal{H}}$ such that the adversary $\mathcal{Z}$ produced the ciphertext using randomizer R', or the ciphertext is a re-randomization of a ciphertext in $\langle \mathtt{C}_{0,j} \rangle_{j \in [n] \setminus \mathbf{C}^S}$. The procedure described in Step 8 successfully recognizes these randomizers. Indeed, for any $\mathtt{C}_{m,j}$, let $\tilde{R}_j$ be the value of its randomizer, and let $\tilde{p} = \mathcal{G}(\tilde{\mathtt{M}}_j \| \mathcal{H}(\tilde{R}_j)))$. If $R' = \tilde{R}_j$, then $p = \mathcal{G}(\tilde{\mathtt{M}}_j \| \mathcal{H}(R'))) = \tilde{p}$, and therefore $\bar{X}_{m,j} + \bar{X}'$ is distributed exactly as a fresh ciphertext and thus $\tilde{R}_j$ is successfully recovered. If $R' \neq \tilde{R}_j$, then with overwhelming probability $p = \mathcal{G}(\tilde{\mathtt{M}}_j \| \mathcal{H}(R'))) \neq \tilde{p}$, and thus the value $\bar{X}_{m,j} + \bar{X}'$ would trigger a decryption error with overwhelming probability. Therefore with overwhelming probability:

$$\textstyle\sum_j [-\mathbf{a}, 1]^T \cdot \bar{W}_{m,j} / (\prod_{l=1}^{m} \kappa_l) = \sum_j \tilde{R}_j \tag{5}$$

By putting together Eq. 3, Eq. 4 and Eq. 5 we have that R^* as computed in step 9 is equal to the randomizer used in the challenge ciphertext. By an argument similar to the one for step (8), the computation of steps (10)–(12) can figure out what message was encrypted by the challenger, i.e., if the decryption oracle answer is not $\perp$ the challenge ciphertext is an encryption of $\mathtt{M}_{i,0}$.

Concluding, conditioned on $\mathtt{Bad} \wedge \mathtt{E}$, B wins the RCCA-security game with overwhelming probability .

By joining the Claim 11 and Claim 12 we have that:

$$\Pr[\mathbf{Exp}^{\mathtt{RCCA}}_{\mathsf{B},\mathcal{PKE}}(\lambda) = 1 \wedge \mathtt{Bad}] \geq \Pr[\mathtt{Bad}] - \mathtt{negl}(\lambda).$$

Using the above bound in Eq. 2 we have that the advantage of B is $\frac{1}{2}\Pr[\mathtt{Bad}] - \mathtt{negl}(\lambda)$, and thus $\Pr[\mathtt{Bad}]$ is negligible. This shows that the two hybrids are indistinguishable.

Hybrid 5. The hybrid experiment $\mathbf{H}_5$ is as the hybrid $\mathbf{H}_4$ but with the following difference:

1. At initialization phase, $\mathsf{flg}_{\mathtt{copied}}$ is initialized to false;
2. When the message $(\mathtt{write}, (\mathcal{L}_m, \mathsf{com}_m))$ appears in the port $\mathtt{BB.in}_{M_m}$, for every $j \in [n]$ compute $\mathtt{M}_{m,j} \leftarrow \mathsf{ADec}(\mathsf{sk}, \prod_{l=1}^{m} \kappa_l, \mathtt{C}_{m,j})$, and define the multiset $\mathcal{V}_m \leftarrow \{\mathtt{M}_{m,j}\}_{j \in [n]}$.
   If $\exists j, j' \in [n]$ such that $j \neq j'$, $\mathtt{M}_{m,j} = \mathtt{M}_{m,j'}$ and $\psi_{\mathsf{hide}}(\mathtt{M}_{m,j}) \in \hat{\mathcal{M}}_H$ then set the flag $\mathsf{flg}_{\mathtt{copied}}$ to true.
3. At the end of the protocol's execution, if $\mathsf{flg}_{\mathtt{copied}}$ is true and a message $(\mathtt{write}, \mathtt{valid})$ appears in a port $\mathtt{BB.in}_{M_i}$ for $i \in [n] \setminus \mathbf{C}^M$ then **abort**;

Lemma 40. $\mathbf{H}_5 \approx_c \mathbf{H}_4$.

The indistinguishability of the two experiments can be reduced to the RCCA-Security of $\mathcal{PKE}$ (as well as the simulation extractability and zero-knowledge of $\mathcal{NIZK}_{\mathtt{sd}}$ and adaptive soundness of $\mathcal{NIZK}_{\mathtt{mx}}$).

The proof is very similar to the one of the previous lemma, and thus here we give only a sketch pointing out the differences. In the proof of lemma 39 the adversary B computes the randomizer $R^* \leftarrow (\sum_j \tilde{R}_j) - (\sum_{j \neq i} R_j)$ in step 9. That was done to bound the event that the challenge ciphertext is not included in the list $\mathcal{L}_m$.

In this hybrid, instead we want to bound the event that the challenge ciphertext appears more than once in $\mathcal{L}_m$. Let $\mathcal{J}$ be the set of indexes such that $j \in \mathcal{J}$ if and only if $\mathtt{C}_{m,j}$ decrypts to $\diamond$. We can build an adversary B that works the same as the adversary of lemma 39 except that the step 9 here becomes:

9. Compute $R^* \leftarrow ((\sum_{j \in [n] \setminus \mathcal{J}} \tilde{R}_j) - (\sum_{j \neq i} R_j))/(1 - |\mathcal{J}|)$.

The distinghuishing condition tells us that $|\mathcal{J}| > 1$, thus the operation above is well defined. Moreover, by the soundness of $\mathcal{NIZK}_{\mathtt{mx}}$ we have that $\sum_{j\in[n]} \tilde{R}_j = \sum_j R_j$ and so the computed randomizer R^* is indeed the randomizer used in the challenge ciphertext $\mathtt{C}^*$, and once R^* is learned it can be used to successfully figure out which of the two plaintexts was encrypted by the challenger.

Hybrid 6. The hybrid experiment $\mathbf{H}_6$ is as the hybrid $\mathbf{H}_5$ but with the following difference:

1. At the end of the protocol's execution, **abort** if $(\mathtt{write}, \mathtt{valid})$ appears in a port $\mathtt{BB.in}_{M_i}$ for $i \in [n] \setminus \mathbf{C}^M$ and $\psi_{\mathsf{hide}}(\mathcal{V}_m) \not\subseteq \mathcal{V}_{h^*}$;

Lemma 41. $\mathbf{H}_6 \approx_c \mathbf{H}_5$.

Proof. The views of $\mathcal{Z}$ in the two two hybrids is exactly the same until the execution reaches the step described above: $\mathbf{H}_6$ might abort while $\mathbf{H}_5$ would continue its execution. Let us call $\mathtt{Forged}$ the event of this aborting condition. By lemma 1, $\Pr[\mathbf{H}_6 = 1] - \Pr[\mathbf{H}_5 = 1] \leq \Pr[\mathtt{Forged}]$. In the following claim we show that $\mathtt{Forged}$ occurs with negligible probability.

Claim 13 $\Pr[\mathtt{Forged}] \in \mathtt{negl}(\lambda)$.

Proof. We prove the claim by doing a reduction to the UF-CCA security of $\mathcal{APKE}$. Below we show an adversary for the UF-CCA security experiment:

Adversary $\mathsf{B}(\mathsf{pk})$:

1. Run the hybrid $\mathbf{H}_5$ (but do not sample the public key of $\mathcal{APKE}$, instead use the value pk received as input) and do not compute the set $\mathcal{V}_0$;
2. When the message $(\mathtt{write}, (\mathcal{L}_{h^*-1}, \mathsf{com}_{h^*-1}))$ appears in the port $\mathtt{OMix.infl}_{M_{h^*-1}}$ set $\tilde{\kappa} \leftarrow \prod_{l=1}^{h^*-1} \kappa_k$ outputs to the challenger the tuple $(\mathcal{L}_{h^*-1}, \tilde{\kappa})$.

Adversary $\mathsf{B}(\{\mathtt{C}'_j\}_{j\in[n]})$:

3. Resume the execution of the hybrid, and for any $j \in [n]$ set the ciphertext $\mathtt{C}_{h^*,\pi_{h^*}(j)} \leftarrow \mathtt{C}'_j$ and let $\mathcal{L}_{h^*} \leftarrow \langle \mathtt{C}_{h^*,j} \rangle_{j\in[n]}$, and sample $\mathsf{com}_{h^*} \leftarrow\!\!\$\ \{0,1\}^\lambda$ (the adversary B does not know the preimage of com_{h^*});
4. Continue the computation until the message $(\mathtt{write}, (\mathcal{L}_m, \mathsf{com}_m))$ appears in the port $\mathtt{BB.out}_{M_m}$;
5. Sample an index $j^* \leftarrow\!\!\$\ [n]$ and output the ciphertext $\mathtt{C}^* \leftarrow \mathtt{C}_{m,j^*} / \prod_{l=h^*+1}^{m} \kappa_l$.

We show that $\Pr[\mathbf{Exp}^{\mathtt{UF-CCA}}_{\mathsf{B},\mathcal{APKE}}(\lambda) = 1] \geq \Pr[\mathtt{Forged}]/n$. First, notice that the adversary B samples $\mathsf{com}_{h^*} \leftarrow\!\!\$\ \{0,1\}^\lambda$ while the hybrid experiment produces it as an hash of κ_{h^*}. However, the preimage of com_{h^*} is never used in the execution of B, so B simulates perfectly (up to where it stops) the view of $\mathcal{Z}$ in $\mathbf{H}_6$. Moreover, κ_{h^*} is uniformly distributed given the view of B, and therefore, it is distributed as the re-randomization key in of the UF-CCA experiment.

Next, conditioning on the event $\mathtt{Forged}$, we have that, $\psi_{\mathsf{hide}}(\mathcal{V}_m) \not\subseteq \mathcal{V}_{h^*}$, the protocol execution is valid and Ψ_{hide} is empty, which means that at least one of the messages in $\mathcal{V}_m$ is not in $\mathcal{V}_{h^*}$. Let j be the index such that $\mathtt{C}_{m,j}$ decrypts, under the authentication key κ_{h^*}, to a message not in $\mathcal{V}_{h^*}$. In the conditional space where $\mathtt{Forged}$ and $j = j^*$ hold, B wins with probability 1. So the claim follows.

Hybrid 7. The hybrid experiment $\mathbf{H}_7$ is as the hybrid $\mathbf{H}_6$ but with the following difference:

– When the message $(\mathtt{write}, (\Pi^{\mathtt{M}}_{h^*}, \kappa_{h^*}))$ appears in the port $\mathtt{BB.in}_{M_{h^*}}$ replace it with $(\mathtt{write}, (\hat{\Pi}^{\mathtt{M}}{}_{h^*}, \kappa_{h^*}))$ where:

$$\hat{\Pi}^{\mathtt{M}} \leftarrow \mathsf{Sim}_{\mathtt{mx}}(\mathsf{td}^s_{\mathtt{mx}}, (h^*, [\mathbf{g}, a], \mathbf{CkSum}(\mathcal{L}_{h^*-1}, \mathcal{L}_{h^*}, \kappa_{h^*}))).$$

Lemma 42. $\mathbf{H}_7 \approx_c \mathbf{H}_6$.

The lemma follows by a simple reduction to the adaptive zero-knowledge property of $\mathcal{NIZK}_{\mathtt{mx}}$, we omit the details.

Hybrid 8. The hybrid experiment $\mathbf{H}_8$ is as the hybrid $\mathbf{H}_7$ but with the following difference:

1. When the message $(\mathtt{write}, (\mathcal{L}_{h^*-1}, \mathsf{com}_{h^*-1}))$ appears in the port $\mathtt{OMix.infl}_{M_{h^*-1}}$, then for all $j \in [n]$ compute the ciphertext $\mathtt{C}_{h^*,\pi(j)}$ as follow:

$$\mathtt{C}_{h^*,\pi(j)} \leftarrow \begin{cases} \mathsf{ARand}(\mathsf{pk}, \prod_{l=1}^{h^*} \kappa_k, \mathsf{Enc}(\mathsf{pk}, \mathtt{M}_{h^*-1,j})) & \text{if } \mathtt{M}_{h^*-1,j} \neq \perp \\ \mathsf{ARand}(\mathsf{pk}, \kappa_{h^*}, \mathtt{C}_{h^*-1,j}) & \text{else} \end{cases}$$

   where $\mathtt{M}_{h^*-1,j} \leftarrow \mathsf{ADec}(\mathsf{sk}, \prod_{l=1}^{h^*-1} \kappa_l, \mathtt{C}_{h^*-1,j})$.
   Set $\mathcal{L}_{h^*} \leftarrow \langle \mathtt{C}_{h^*,j} \rangle_{j\in[n]}$ and continue the execution of the experiment.

Lemma 43. $\mathbf{H}_8 \approx_c \mathbf{H}_7$.

Proof. To do the proof we use a sequence of intermediate hybrids where the condition above is weakened. Specifically let $\mathbf{H}_{8,i}$ be the same as $\mathbf{H}_8$ but where the step 1 is modified as follow:

1. When the message $(\mathtt{write}, (\mathcal{L}_{h^*-1}, \mathsf{com}_{h^*-1}))$ appears in the port $\mathtt{OMix.infl}_{M_{h^*-1}}$, <u>for all $1 \leq j < i$</u> compute the ciphertext $\mathtt{C}_{h^*,\pi(j)}$ as follow:

$$\mathtt{C}_{h^*,\pi(j)} \leftarrow \begin{cases} \mathsf{ARand}(\mathsf{pk}, \prod_{l=1}^{h^*} \kappa_l, \mathsf{Enc}(\mathsf{pk}, \mathtt{M}_{h^*-1,j})) & \text{if } \mathtt{M}_{h^*-1,j} \neq \perp \\ \mathsf{ARand}(\mathsf{pk}, \kappa_{h^*}, \mathtt{C}_{h^*,j}) & \text{else} \end{cases}$$

   where $\mathtt{M}_{h^*-1,j} \leftarrow \mathsf{ADec}(\mathsf{sk}, \prod_{l=1}^{h^*-1} \kappa_l, \mathtt{C}_{h^*-1,j})$.
   <u>For all $i \leq j \leq n$ compute the ciphertext $\mathtt{C}_{h^*,\pi(j)}$ as in $\mathbf{H}_7$.</u>
   Set $\mathcal{L}_{h^*} \leftarrow \langle \mathtt{C}_{h^*,j} \rangle_{j\in[n]}$ and continue the execution of the experiment.

   Notice that $\mathbf{H}_{8,1} \equiv \mathbf{H}_7$ and $\mathbf{H}_{8,n+1} \equiv \mathbf{H}_8$.

Claim 14 *For all $i \in [n]$, $\mathbf{H}_{8,i+1} \approx_c \mathbf{H}_{8,i}$.*

Proof. The hybrids differ because $\mathbf{H}_{8,i+1}$ would decrypt and re-encrypt $\mathtt{C}_{h^*-1,i}$, when its decryption is not $\perp$, while $\mathbf{H}_{8,i}$ would always re-randomize it. Below we show how to reduce the indistinguishability of these hybrids to the weak-RCCA re-randomizability of $\mathcal{APKE}$.[7] Consider the following adversary:

<u>Adversary $\mathsf{B}(\mathsf{pk})$</u>:

1. Run the hybrid $\mathbf{H}_{8,i}$ (but do not sample the public key of $\mathcal{PKE}$, instead use the value pk received as input) and compute $\mathcal{V}_0$ (the list of plaintexts of $\mathcal{L}_0$) using the decryption oracle Dec.
2. When the message $(\mathtt{write}, (\mathcal{L}_{h^*-1}, \mathsf{com}_{h^*-1}))$ appears in the port $\mathtt{OMix.infl}_{M_{h^*-1}}$, compute the ciphertext $\mathtt{C}' \leftarrow \mathtt{C}_{h^*-1,i} / \prod_{l=1}^{h^*-1} \kappa_l$, and query the oracle Dec on $\mathtt{C}'$; let $\tilde{\mathtt{M}}$ be its answer and assume that $\tilde{\mathtt{M}} \neq \perp$ (in fact, when the i-th ciphertext decrypts to $\perp$ the hybrids are identical and thus we want to make a reduction only for the other case). Next, send $\mathtt{C}'$ to the challenger, receive a ciphertext $\mathtt{C}^*$, and set $\mathtt{C}_{h^*,\pi_{h*}(i)} \leftarrow \mathsf{ARand}(\mathsf{pk}, \prod_{l=1}^{h^*} \kappa_l, \mathtt{C}^*)$. From now on, B has access to the oracle $\mathsf{Dec}^{\perp}$.

[7] A reduction to (full-fledged) RCCA re-randomizability would be straightforward; however, our scheme is only weak-RCCA re-randomizable, therefore we need to work a bit harder to make the reduction works.

3. Sample an index $i^* \leftarrow_{\$} [n+1]$. To simulate the decode phase (step 5) of the protocol, for all $j \in [n]$ decrypt the ciphertext $\mathtt{C}_{m,j}$ using the oracle $\mathsf{Dec}^{\perp}$; specifically, by sending $\mathtt{C}_{m,j}/\prod_{l=1}^{m} \kappa_l$. If the answer is $\perp$ and $j < i^*$, then replace it with $\tilde{\mathtt{M}}$; else if the answer is $\perp$ and $j \geq i^*$, answer the decryption with $\perp$. Notice that in the latter case the protocol enters the `invalid` branch (see next step). Here i^* is a guess for the event that the i^*-th decryption query is the first for which $\mathsf{Dec}^{\perp}$ outputs $\perp$ while it is actually the case that the queried ciphertext decrypts to $\perp$. $i^* = n+1$ corresponds to the case where none of these queries falls in this case.
4. Continue the execution of $\mathbf{H}_{8,i}$, in case the protocol turns to the `invalid` branch, then it will decrypts the ciphertext of $\mathcal{L}_0$, uses the decryption of $\mathcal{V}_0$ to answer all the subsequent decryptions, and finally outputs the bit that $\mathcal{Z}$ outputs.

We define the following events:

- Let `Guess` be the event that the i^*-th decryption query is the first for which $\mathsf{Dec}^{\perp}$ outputs $\perp$ and, actually, $\mathsf{Dec}(\mathsf{sk}, \mathtt{C}_{m,i^*}) = \perp$.
- Let `Rand` be the event that $\mathsf{Dec}(\mathsf{sk}, \mathtt{C}') \neq \perp$.
- Let `Good` be the event `Guess` $\wedge$ `Rand`.

Claim 15 *For $i \in [n]$, $\Pr[\mathbf{H}_{8,i} = 1 \mid \neg\mathtt{Rand}] = \Pr[\mathbf{H}_{8,i+1} = 1 \mid \neg\mathtt{Rand}]$*

Proof. If the event `Rand` does not happen then both the hybrids will re-randomize the ciphertext $\mathtt{C}_{h^*-1,i}$ following the procedure defined by the Π. Therefore, there isn no difference between the two hybrids.

We show that conditioning on the event `Good` the behavior of B in $\mathbf{Exp}^{\mathtt{Rand-wRCCA}}$ perfectly matches the hybrids $\mathbf{H}_{8,i+1}$ and $\mathbf{H}_{8,i}$ (depending on the challenge bit of the experiment).

Claim 16 *For $i \in [n]$, $\Pr[\mathbf{H}_{8,i} = 1 \mid \mathtt{Rand}] = \Pr[\mathbf{Exp}^{\mathtt{Rand-wRCCA}}_{\mathsf{B},\mathcal{PKE}}(\lambda) = 1 \mid b^* = 1,\ \mathtt{Good}]$.*

Proof. Since the challenge bit $b^* = 1$ then the ciphertext $\mathtt{C}_{h^*,\pi_{h^*}(i)}$ is computed as:

$$\mathsf{ARand}(\mathsf{pk}, \textstyle\prod_{l=1}^{h^*} \kappa_l, \mathsf{Rand}(\mathsf{pk}, \mathtt{C}_{h^*-1,i}/\prod_{l=1}^{h^*-1} \kappa_l)),$$

which, by a simple inspection of the re-randomization algorithms, is equivalent to $\mathsf{ARand}(\mathsf{pk}, \kappa_{h^*}, \mathtt{C}_{h^*-1,i})$, and thus the distribution of $\mathcal{L}_{h^*}$ is the same in B's simulation and in $\mathbf{H}_{8,i}$. However, we have not considered the decryption oracle queries yet. Notice that we are conditioning on `Guess`, hence the behavior of B exactly matches the pattern of decryptions for all the queries answered with $\perp$ by the decryption oracle, therefore B perfectly simulates $\mathbf{H}_{8,i}$ when $b^* = 1$.

Claim 17 *For $i \in [n]$, $\Pr[\mathbf{H}_{8,i+1} = 1 \mid \mathtt{Rand}] = \Pr[\mathbf{Exp}^{\mathtt{Rand-wRCCA}}_{\mathsf{B},\mathcal{PKE}}(\lambda) = 0 \mid b^* = 0,\ \mathtt{Good}]$.*

Proof. As in the proof of the previous claim, since we condition on `Guess` the behavior of B exactly matches the pattern of decryptions. Moreover, by inspection the procedure to decrypt and re-encrypt executed by B in the rand-wRCCA experiment and in the hybrid $\mathbf{H}_{8,i+1}$ are exactly the same.

Let $B := \mathbf{Exp}^{\mathtt{Rand-wRCCA}}_{\mathsf{B},\mathcal{PKE}}(\lambda) = 1$, by easy calculations:

$$\begin{aligned}
&\Pr[\mathbf{Exp}^{\mathtt{Rand-wRCCA}}_{\mathsf{B},\mathcal{PKE}}(\lambda) = 1] \\
&= \Pr[B \mid \mathtt{Good}] \cdot \Pr[\mathtt{Good}] + \Pr[B \mid \neg\mathtt{Good}] \cdot \Pr[\neg\mathtt{Good}] \\
&\geq \Pr[B \mid \mathtt{Good}] \cdot \Pr[\mathtt{Good}] + \tfrac{1}{2} \cdot \Pr[\neg\mathtt{Good}] \\
&= \tfrac{1}{2}\left(\Pr[B \mid b^* = 0, \mathtt{Good}] + \Pr[B \mid b^* = 1, \mathtt{Good}]\right) \cdot \Pr[\mathtt{Good}] + \tfrac{1}{2} \cdot \Pr[\neg\mathtt{Good}] \\
&= \tfrac{1}{2}\left(1 - \Pr[\neg B \mid b^* = 0, \mathtt{Good}] + \Pr[B \mid b^* = 1, \mathtt{Good}]\right) \cdot \Pr[\mathtt{Good}] + \tfrac{1}{2} \cdot \Pr[\neg\mathtt{Good}] \\
&= \tfrac{1}{2}(\Pr[\mathtt{Good}] + \Pr[\neg\mathtt{Good}]) + (\Pr[\mathbf{H}_{8,i+1} = 1] - \Pr[\mathbf{H}_{8,i} = 1])/2 \cdot \Pr[\mathtt{Good}] \\
&= \tfrac{1}{2} + (\Pr[\mathbf{H}_{8,i+1} = 1 \mid \mathtt{Rand}] - \Pr[\mathbf{H}_{8,i} = 1 \mid \mathtt{Rand}])/2 \cdot \Pr[\mathtt{Good}] \,.
\end{aligned}$$

Finally, notice that the probability of $\mathtt{Guess}$ is $1/(n+1)$ (which is non-negligible in λ), independently of the probability of $\mathtt{Rand}$ therefore, putting together the derivation above with the Claim 15 and by the rand-wRCCA security of $\mathcal{PKE}$ the two hybrids are indistinguishable.

Hybrid 9. The hybrid experiment $\mathbf{H}_9$ is as the hybrid $\mathbf{H}_8$ but with the following difference:

1. At the initialization phase sample another set of inputs $\mathcal{M}_{\mathsf{hide}} \leftarrow \{\mathtt{H}_j\}_{j\in[n]}$ such that $\mathtt{H}_j \leftarrow_{\$} \mathbb{G}^\ell$, $\mathcal{M}_{\mathsf{hide}}$ contains no repetitions, and $\mathcal{M}_{\mathsf{hide}} \cap \hat{\mathcal{M}}_H = \emptyset$.
2. When the message $(\mathtt{write}, (\mathcal{L}_{h^*-1}, \mathsf{com}_{h^*-1}))$ appears in the port $\mathtt{OMix.infl}_{M_{h^*-1}}$, for all $j \in [n]$, compute $\mathtt{M}_{h^*-1,j} \leftarrow \mathsf{ADec}(\mathsf{sk}, \prod_{l=1}^{h^*-1} \kappa_l, \mathtt{C}_{h^*-1,j})$ and do the following:
   - Compute the ciphertext $\mathtt{C}_{h^*,\pi(j)}$ as follow:

$$\mathtt{C}_{h^*,\pi_{h^*}(j)} \leftarrow \begin{cases} \mathsf{ARand}(\mathsf{pk}, \prod_{l=1}^{h^*} \kappa_l, \mathsf{Enc}(\mathsf{pk}, \underline{\mathtt{H}_j})) & \text{if } \mathtt{M}_{h^*-1,j} \neq \perp \\ \mathsf{ARand}(\mathsf{pk}, \kappa_{h^*}, \mathtt{C}_{h^*-1,j}) & \text{else} \end{cases}$$

   - If $\mathtt{M}_{h^*-1,j} \neq \perp$ then add $(\mathtt{H}_j, \mathtt{M}_{h^*-1,j})$ to Ψ_{hide}.

   Set $\mathcal{L}_{h^*} \leftarrow \langle \mathtt{C}_{h^*,j} \rangle_{j\in[n]}$ and continue the execution of the experiment.

Lemma 44. $\mathbf{H}_9 \approx_c \mathbf{H}_8$.

Proof. We prove indistinguishability based on the RCCA-security of $\mathcal{PKE}$. To do so we define intermediate hybrids: let $\mathbf{H}_{9,i}$ be the experiment that re-encrypts the first i ciphertexts as in $\mathbf{H}_9$, and all the other ciphertexts as in $\mathbf{H}_8$. More precisely, $\mathbf{H}_{9,i}$ is as $\mathbf{H}_9$ except that step 2 is modified as follow:

2. When the message $(\mathtt{write}, (\mathcal{L}_{h^*-1}, \mathsf{com}_{h^*-1}))$ appears in the port $\mathtt{OMix.infl}_{M_{h^*-1}}$:
   - For all $j < i$ compute the ciphertext $\mathtt{C}_{h^*,\pi_{h^*}(j)}$ as in $\mathbf{H}_9$ (i.e., re-randomize a fresh encryption of $\mathtt{H}_j$ and add $(\mathtt{H}_j, \mathtt{M}_{h^*-1,j})$ to Ψ_{hide});
   - For all $i \leq j \leq n$ compute the ciphertext $\mathtt{C}_{h^*,\pi_{h^*}(j)}$ as in $\mathbf{H}_8$.

   Set $\mathcal{L}_{h^*} \leftarrow \{\mathtt{C}_{h^*,j}\}_{j\in[n]}$ and continue the execution of the experiment.

Clearly, $\mathbf{H}_{9,1} \equiv \mathbf{H}_8$ and $\mathbf{H}_{9,n+1} \equiv \mathbf{H}_9$. Below we show that $\mathbf{H}_{9,i} \approx_c \mathbf{H}_{9,i+1}$ for all $i \in [n]$. Let $\mathtt{Rand}$ be the event that $\mathsf{ADec}(\mathsf{sk}, \prod_{l=1}^{h^*-1} \kappa_l, \mathtt{C}_{h^*-1,i}) \neq \perp$.

Claim 18 *For $i \in [n]$, $\Pr[\mathbf{H}_{9,i} = 1 \mid \neg\mathtt{Rand}] = \Pr[\mathbf{H}_{9,i+1} = 1 \mid \neg\mathtt{Rand}]$.*

The claim follows by inspection: by hybrid definitions one can see that when $\mathtt{Rand}$ does not happen both hybrids re-randomize the ciphertexts following the procedure described in Π.

Next, consider the following adversary for the RCCA-security experiment:

Adversary $\mathsf{B}(\mathsf{pk})$:

1. Run the hybrid $\mathbf{H}_{9,i}$ (but do not sample the public key of $\mathcal{PKE}$, instead use the value pk received as input); (Pick a set $\mathcal{M}_{\mathsf{hide}} \leftarrow \{\mathtt{H}_j\}_{j\in[n]}$ where $\mathtt{H}_j \leftarrow_{\$} \mathbb{G}^\ell$, all the messages in the set $\mathcal{M}_{\mathsf{hide}}$ are distinct, and $\mathcal{M}_{\mathsf{hide}} \cap \hat{\mathcal{M}}_H = \emptyset$;)
2. When the message $(\mathtt{write}, (\mathcal{L}_{h^*-1}, \mathsf{com}_{h^*-1}))$ appears in the port $\mathtt{OMix.infl}_{M_{h^*-1}}$:
   (a) For all $j \leq i-1$ compute the ciphertext $\mathtt{C}_{h^*,\pi_{h^*}(j)}$ as in $\mathbf{H}_9$. and add $(\mathtt{H}_j, \mathtt{M}_{h^*-1,j})$ to ψ_{hide};
   (b) Query the decryption oracle on $\mathtt{C}' \leftarrow (\mathtt{C}_{h^*-1,i}/\prod_{l=1}^{h^*-1}\kappa_l)$; let $\mathtt{M}_{h^*-1,i}$ be its response;
   (c) Send the pair $(\mathtt{M}_{h^*-1,i}, \mathtt{H}_i)$ to the challenger, and let $\mathtt{C}^*$ be the challenge ciphertext received in reponse. Next, set $\mathtt{C}_{h^*,\pi_{h^*}(i)} \leftarrow \mathsf{ARand}(\mathsf{pk}, \prod_{l=1}^{h^*}\kappa_l, \mathtt{C}^*)$ and add $(\mathtt{H}_i, \mathtt{M}_{h^*-1,i})$ to Ψ_{hide}. Moreover, add $(\diamond, \psi_{\mathsf{in}}(\mathtt{M}_{h^*-1,i}))$ to Ψ_{in}; this latter assignment implies that if $\mathtt{M}_{h^*-1,i} \in \hat{\mathcal{M}}_H$, then $\diamond$ is mapped to the corresponding honest sender input, otherwise the assignment is equivalent to adding $(\diamond, \mathtt{M}_{h^*-1,i})$ to Ψ_{in}.)
   (d) For all $i < j \leq n$ compute the ciphertext $\mathtt{C}_{h^*,\pi_{h^*}(j)}$ as in $\mathbf{H}_8$.
3. Compute all the decryptions that the hybrid $\mathbf{H}_{9,i}$ performs (for example, to compute $\mathcal{V}_{h^*}, \mathcal{V}_m$) with the help of the oracle $\mathsf{Dec}^\diamond$ and whenever the decryption oracle outputs $\diamond$ replace it with $\mathtt{M}_{h^*-1,i}$;
4. At the end of the hybrid experiment outputs what $\mathcal{Z}$ outputs.

Claim 19 *The following claims are both true:*

1. $\Pr[\mathbf{H}_{9,i} = 1 \mid \mathtt{Rand}] \approx_s \Pr[\mathbf{Exp}^{\mathtt{RCCA}}_{\mathsf{B},\mathcal{PKE}}(\lambda) = 0 \mid b^* = 0,\ \mathtt{Rand}]$,
2. $\Pr[\mathbf{H}_{9,i+1} = 1 \mid \mathtt{Rand}] = \Pr[\mathbf{Exp}^{\mathtt{RCCA}}_{\mathsf{B},\mathcal{PKE}}(\lambda) = 0 \mid b^* = 1,\ \mathtt{Rand}]$.

Proof. The adversary B perfectly simulates all the step of $\mathbf{H}_{9,i}$ until the challenge message is sent. At this point, if the challenge bit $b^* = 0$ then the adversary receives an encryption $\mathtt{C}^*$ of the message $\mathtt{M}_{h^*-1,i}$ and therefore the ciphertext $\mathtt{C}_{h^*,\pi_{h^*}(i)}$ is distributed exactly as in $\mathbf{H}_{9,i}$. On the other hand, if the challenge bit is 1 then the adversary receives an encryption $\mathtt{C}^*$ of the message $\mathtt{H}_i$, and therefore the ciphertext $\mathtt{C}_{h^*,\pi_{h^*}(i)}$ is distributed exactly as in $\mathbf{H}_{9,i+1}$.

Notice that the hybrid $\mathbf{H}_{9,i}$ does not add $(\mathtt{H}_i, \mathtt{M}_{h^*-1,i})$ to Ψ_{hide} while B does it. This may raise a difference in B's simulation. However, the value $\mathtt{H}_i$ is uniformly random over a set of size at least $|\mathbb{G}^\ell| - 2n$, and the view of $\mathcal{Z}$ is independent of $\mathtt{H}_i$. Hence, the simulation can diverge only if $\mathcal{Z}$ guesses the value $\mathtt{H}_i$ which can happen only with negligible probability.

By simple inspection of step 3 of B, we notice that B and $\mathbf{H}_{9,i}$ compute the sets $\mathcal{V}_{h^*}$ and $\mathcal{V}_m$ in the same way. On the other hand, the two sets are computed differently by B and $\mathbf{H}_{9,i+1}$. In fact, in step 3 of B whenever $\diamond$ occurs B adds $\mathtt{M}_{h^*-1,i}$ to the computed set (either $\mathcal{V}_{h^*}$ or $\mathcal{V}_m$), while $\mathbf{H}_{9,i+1}$ would add $\mathtt{H}_i$. We argue this change does not create any difference in the view of $\mathcal{Z}$. In particular, in $\mathbf{H}_4$ we introduced the check $\hat{\mathcal{M}}_H \not\subseteq \psi_{\mathsf{hide}}(\mathcal{V}_m)$ and in $\mathbf{H}_6$ we introduced the check $\psi_{\mathsf{hide}}(\mathcal{V}_m) \not\subseteq \mathcal{V}_{h^*}$. Notice that $\psi_{\mathsf{hide}}(\mathcal{V}_m)$ as computed by B would map $\mathtt{M}_{h^*-1,i}$ to $\mathtt{M}_{h^*-1,i}$ while in hybrid $\mathbf{H}_{9,i+1}$ would map $\mathtt{H}_i$ to $\mathtt{M}_{h^*-1,i}$. Hence, after applying ψ_{hide}, the checks work with identical sets. Finally, concerning the check of $\mathbf{H}_5$ – $\psi_{\mathsf{hide}}(\mathtt{M}_{m,j}) \in \hat{\mathcal{M}}_H$ – for B if $\mathtt{M}_{m,j} = \diamond$ this is remapped to $\mathtt{M}_{h^*-1,i}$ while in $\mathbf{H}_{9,i+1}$ would be $\mathtt{H}_i$: for the same reason as above, after applying ψ_{hide} the check computed in B's simulation works the same as in $\mathbf{H}_{9,i+1}$.

Let $B := \mathbf{Exp}^{\mathtt{Rand-wRCCA}}_{\mathsf{B},\mathcal{PKE}}(\lambda) = 1$, by easy calculations:

$$\begin{aligned}
&\Pr[\mathbf{Exp}^{\mathtt{Rand-wRCCA}}_{\mathsf{B},\mathcal{PKE}}(\lambda) = 1] \\
&= \Pr[B \mid \mathtt{Rand}] \cdot \Pr[\mathtt{Rand}] + \Pr[B \mid \neg\mathtt{Rand}] \cdot \Pr[\neg\mathtt{Rand}] \\
&\geq \Pr[B \mid \mathtt{Rand}] \cdot \Pr[\mathtt{Rand}] + \tfrac{1}{2} \cdot \Pr[\neg\mathtt{Rand}] \\
&= \tfrac{1}{2}\left(\Pr[B \mid b^* = 0, \mathtt{Rand}] + \Pr[B \mid b^* = 1, \mathtt{Rand}]\right) \cdot \Pr[\mathtt{Rand}] + \tfrac{1}{2} \cdot \Pr[\neg\mathtt{Rand}] \\
&= \tfrac{1}{2}\left(1 - \Pr[\neg B \mid b^* = 0, \mathtt{Rand}] + \Pr[B \mid b^* = 1, \mathtt{Rand}]\right) \cdot \Pr[\mathtt{Rand}] + \tfrac{1}{2} \cdot \Pr[\neg\mathtt{Rand}] \\
&= \tfrac{1}{2}(\Pr[\mathtt{Rand}] + \Pr[\neg\mathtt{Rand}]) + (\Pr[\mathbf{H}_{9,i+1} = 1] - \Pr[\mathbf{H}_{9,i} = 1])/2 \cdot \Pr[\mathtt{Rand}] \\
&= \tfrac{1}{2} + (\Pr[\mathbf{H}_{9,i+1} = 1 \mid \mathtt{Rand}] - \Pr[\mathbf{H}_{9,i} = 1 \mid \mathtt{Rand}])/2 \cdot \Pr[\mathtt{Rand}] .
\end{aligned}$$

The lemma follows by putting together Claim 18 and the derivation above, and by the RCCA-security of $\mathcal{PKE}$.

Hybrid 10. The hybrid experiment $\mathbf{H}_{10}$ is as the hybrid $\mathbf{H}_9$ but with the following difference:

1. When the message $(\mathtt{write}, (\mathcal{L}_{h^*-1}, \mathsf{com}_{h^*-1}))$ appears in the port $\mathtt{OMix.infl}_{M_{h^*-1}}$, for all $j \in [n]$, compute $\mathtt{M}_{h^*-1,j} \leftarrow \mathsf{ADec}(\mathsf{sk}, \prod_{l=1}^{h^*-1} \kappa_l, \mathtt{C}_{h^*-1,j})$ and do the following:
   - Compute the ciphertext

$$\mathtt{C}_{h^*,\pi_{h^*}(j)} \leftarrow \begin{cases} \mathsf{ARand}(\mathsf{pk}, \prod_{l=1}^{h^*} \kappa_l, \mathsf{Enc}(\mathsf{pk}, \underline{\mathtt{H}_{\pi_{h^*}(j)}})) & \text{if } \mathtt{M}_{h^*-1,j} \neq \perp \\ \mathsf{ARand}(\mathsf{pk}, \kappa_{h^*}, \mathtt{C}_{h^*-1,j}) & \text{else} \end{cases}$$

   - If $\mathtt{M}_{h^*-1,j} \neq \perp$ then add $(\underline{\mathtt{H}_{\pi_{h^*}(j)}}, \mathtt{M}_{h^*-1,j})$ to Ψ_{hide}.

Claim 20 $\mathbf{H}_{10} \equiv \mathbf{H}_9$.

Proof. There are two syntactical differences between $\mathbf{H}_{10}$ and $\mathbf{H}_9$ (underlined above).

The first one is that $\mathtt{C}_{h^*,\pi_{h^*}(j)}$ is assigned the encryption of $\mathtt{H}_{\pi_{h^*}(j)}$ in $\mathbf{H}_{10}$, and the encryption of $\mathtt{H}_j$ in $\mathbf{H}_9$. However, for any $j \in [n]$, the distributions of $\mathtt{H}_j$ and $\mathtt{H}_{\pi_{h^*}(j)}$ are exactly the same up to the decoding phase, so the list $\mathcal{L}_{h^*}$ has the same distribution in both hybrids (up to the decoding phase). The second difference is that, for all $j \in [n]$ such that $\mathtt{M}_{h^*-1,j} \neq \perp$, $\mathbf{H}_{10}$ adds $(\mathtt{H}_{\pi_{h^*}(j)}, \mathtt{M}_{h^*,j})$ to Ψ_{hide}, whereas $\mathbf{H}_9$ adds $(\mathtt{H}_j, \mathtt{M}_{h^*-1,j})$. The latter implies that, even after the decoding phase, the two lists of ciphertexts are distributed exactly the same. Indeed, both hybrids would return the same value as decryption of the ciphertext $\mathtt{C}_{h^*,j}$.

We are ready now to describe the simulator.

Simulator $\mathsf{S}()$:

Initialization Phase:

1. Set the following flags $\mathsf{flg}_{\mathsf{RO}} \leftarrow \mathsf{false}$, $\mathsf{flg}_{\mathtt{stuffed}} \leftarrow \mathsf{false}$, $\mathsf{flg}_{\mathtt{copied}} \leftarrow \mathsf{false}$.
2. Sample the list of simulated inputs of the honest senders $\hat{\mathcal{M}}_H \leftarrow \{\hat{\mathtt{M}}_j\}_{j \in [n] \setminus \mathbf{C}^S}$ such that each $\hat{\mathtt{M}}_j \leftarrow\!\!\$\ \mathbb{G}^\ell$ is sampled uniformly at random and $\hat{\mathcal{M}}_H$ contains no repetitions.
3. Sample another set of inputs $\mathcal{M}_{\mathsf{hide}} \leftarrow \{\mathtt{H}_j\}_{j \in [n]}$ such that $\mathtt{H}_j \leftarrow\!\!\$\ \mathbb{G}^\ell$, $\mathcal{M}_{\mathsf{hide}}$ contains no repetitions, and $\mathcal{M}_{\mathsf{hide}} \cap \hat{\mathcal{M}}_H = \emptyset$.
4. Run internally the open agent $(\Pi \circ \mathcal{F}'^{\mathsf{Dec}} \circ \mathcal{F}^{\mathsf{NPRO}} \circ \mathrm{BB})$, where the functionality $\mathcal{F}'^{\mathsf{Dec}}$ works the same as $\mathcal{F}^{\mathsf{Dec}}$ except that:
   - Instead of step 2, compute $\mathsf{crs}_{\mathtt{sd}}, \mathsf{td}^s_{\mathtt{sd}}, \mathsf{td}^e_{\mathtt{sd}} \leftarrow \overline{\mathsf{Init}_{\mathtt{sd}}}(1^\lambda)$;
   - Instead of step 3, compute $\mathsf{crs}_{\mathtt{mx}}, \mathsf{td}^s_{\mathtt{mx}} \leftarrow \overline{\mathsf{Init}_{\mathtt{mx}}}(1^\lambda)$.

Listen to all the ports of the open agent system and react as described below.

INPUT SUBMISSION:

5. Forward all the messages from/to the ports $\texttt{OMix.lk}_{S_i}$ and $\texttt{OMix.infl}_{S_i}$ for $i \in \mathbf{C}^S$ of the simulator from/to the same ports in the internal execution of the agent.
6. Whenever a message $(\texttt{input}, j)$ appears in the input port $\texttt{OMix.lk}$ of the simulator, with $j \in [n] \setminus \mathbf{C}^S$ (i.e., the notification that the honest sender $\mathcal{P}_{S_j}$ submitted its input), instead of running the code prescribed by Π, run the following:
   - Compute $\hat{\mathtt{C}}_j \leftarrow \mathsf{Enc}(\mathsf{pk}, \hat{\mathtt{M}}_j; R_j, w_j, x_j, y_j)$ where $R_j \leftarrow\!\!\$\ \mathbb{G}$, $w_j, x_j, y_j \leftarrow\!\!\$\ \mathbb{Z}_q^3$;
   - Compute $\Pi_j^{\mathtt{s}} \leftarrow\!\!\$\ \mathsf{P}_{\mathsf{sd}}(\mathsf{crs}_{\mathsf{sd}}, (j, [\mathbf{g}, a], \bar{W}_j), (w_j, R_j))$ where $\bar{W}_j \leftarrow (w_j \cdot [\mathbf{g}], w_j \cdot [a] + R_j)$;
   - Insert the message $(\texttt{input}, j, (\hat{\mathtt{C}}_j, \Pi_j^{\mathtt{s}}))$ to the port $\texttt{BB.in}_{S_j}$;

OPTIMISTIC MIX:

7. Forward all the messages from/to the ports $\texttt{OMix.lk}_{M_i}$ and $\texttt{OMix.infl}_{M_i}$ for $i \in \mathbf{C}^M$ of the simulator from/to the same ports in the internal execution of the agent.
8. For any $i \in \mathbf{C}^M$ when the message $(\texttt{write}, (\mathcal{L}_i, \mathsf{com}_i))$ appears in the port $\texttt{BB.in}_{M_i}$ check if there exists an entry $(\kappa_i, \mathsf{com}_i) \in \mathcal{Q}^{\mathcal{F}}$: if not, set $\mathsf{flg}_{\mathsf{RO}} \leftarrow \mathsf{true}$;
9. When the message $(\texttt{write}, (\mathcal{L}_{h^*-1}, \mathsf{com}_{h^*-1}))$ appears in the port $\texttt{BB.in}_{M_{h^*-1}}$, sample $\kappa_{h^*} \leftarrow\!\!\$\ \mathbb{Z}_q$, $\pi_{h^*} \leftarrow\!\!\$\ \mathcal{S}_n$, parse $\mathcal{L}_{h^*-1} = \langle \mathtt{C}_{h^*-1,j} \rangle_{j \in [n]}$ and for all $j \in [n]$:
   - Compute $\mathtt{M}_{h^*-1,j} \leftarrow \mathsf{ADec}(\mathsf{sk}, \prod_{l=0}^{h^*-1} \kappa_l, \mathtt{C}_{h^*-1,j})$
   - Compute the ciphertext:

$$\mathtt{C}_{h^*,\pi_{h^*}(j)} \leftarrow \begin{cases} \mathsf{ARand}(\mathsf{pk}, \prod_{l=1}^{h^*} \kappa_l, \mathsf{Enc}(\mathsf{pk}, \mathtt{H}_{\pi_{h^*}(j)})) & \text{if } \mathtt{M}_{h^*-1,j} \neq \perp \\ \mathsf{ARand}(\mathsf{pk}, \kappa_{h^*}, \mathtt{C}_{h^*-1,j}) & \text{else} \end{cases}$$

   - If $\mathtt{M}_{h^*-1,j} \neq \perp$ then add $(\mathtt{H}_{\pi_{h^*}(j)}, \mathtt{M}_{h^*-1,j})$ to Ψ_{hide}.

   Build the multiset $\mathcal{V}_{h^*-1} \leftarrow \{\mathtt{M}_{h^*-1,j} : j \in [n]\}$, define $\mathcal{L}_{h^*} \leftarrow \langle \mathtt{C}_{h^*,j} \rangle_{j \in [n]}$ and continue the execution of the experiment.
10. When the message $(\texttt{write}, (\mathcal{L}_m, \mathsf{com}_m))$ appears in the port $\texttt{BB.in}_{M_m}$, for every $j \in [n]$ compute $\mathtt{M}_{m,j} \leftarrow \mathsf{ADec}(\mathsf{sk}, \prod_{l=1}^{m} \kappa_l, \mathtt{C}_{m,j})$, and define the multiset $\mathcal{V}_m \leftarrow \{\mathtt{M}_{m,j}\}_{j \in [n]}$. Moreover:
    (a) If $\hat{\mathcal{M}}_H \not\subseteq \psi_{\mathsf{hide}}(\mathcal{V}_m)$ then set the flag $\mathsf{flg}_{\texttt{stuffed}}$ to true.
    (b) if $\exists j, j' \in [n]$ such that $j \neq j'$, $\mathtt{M}_{m,j} = \mathtt{M}_{m,j'}$ and $\psi_{\mathsf{hide}}(\mathtt{M}_{m,j}) \in \hat{\mathcal{M}}_H$ then set the flag $\mathsf{flg}_{\mathsf{copied}}$ to true.
11. When the message $(\texttt{write}, (\Pi_{h^*}^{\mathtt{M}}, \kappa_{h^*}))$ appears in the port $\texttt{BB.in}_{M_{h^*}}$ replace it with $(\texttt{write}, (\hat{\Pi}^{\mathtt{M}}{}_{h^*}, v\kappa_{h^*}))$ where
    $\hat{\Pi}^{\mathtt{M}}{}_{h^*} \leftarrow \mathsf{Sim}_{\mathtt{mx}}(\mathsf{td}_{\mathtt{mx}}, (h^*, [\mathbf{g}, a], \mathbf{CkSum}(\mathcal{L}_{h^*-1}, \mathcal{L}_{h^*}, \kappa_{h^*})))$.

VERIFICATION PHASE:

12. If the message $(\texttt{write}, \texttt{valid})$ appears in a port $\texttt{BB.in}_{M_i}$ where $i \in [n] \setminus \mathbf{C}^M$ and $\perp \notin \mathcal{V}_m$ then send the message $\texttt{invalidate}$ in the influence port $\texttt{OMix.infl}$.

DECODE PHASE:

13. If the message $(\texttt{write}, \texttt{valid})$ appears in a port $\texttt{BB.in}_{M_i}$ where $i \in [n] \setminus \mathbf{C}^M$ and one of the following conditions hold then **abort**:
    (a) the flag $\mathsf{flg}_{\mathsf{RO}}$ is true
    (b) the flag $\mathsf{flg}_{\texttt{stuffed}}$ is true;
    (c) the flag $\mathsf{flg}_{\texttt{copied}}$ is true;
    (d) $\psi_{\mathsf{hide}}(\mathcal{V}_m) \not\subseteq \mathcal{V}_{h^*}$;
14. If the message $(\texttt{write}, \texttt{valid})$ appeared then send the following messages to the port $\texttt{OMix.infl}$:
    - For all $j \in [n]$ compute $\mathtt{M}_{m,j} \leftarrow \mathsf{ADec}(\mathsf{sk}, \prod_{l=1}^{m} \kappa_l, \mathtt{C}_{m,j})$ if $\psi_{\mathsf{hide}}(\mathtt{M}_{m,j}) \notin \hat{\mathcal{M}}_H$ then send the message $(\texttt{input}, j, \mathtt{M}_{m,j})$;

– send the message $\texttt{mixDone}$;

Successively, receive the message $(\texttt{mixDone}, \texttt{valid}, \mathcal{O})$. Let $\mathcal{O}^C \leftarrow \psi_{\mathsf{hide}}(\mathcal{V}_m) \setminus \hat{\mathcal{M}}_H$, and compute $\mathcal{O}^H \leftarrow \mathcal{O} \setminus \mathcal{O}^C$ (the ordered list of the honest inputs). Let $\{i_1, \ldots, i_{n-|\mathbf{C}^S|}\} = [n] \setminus \mathbf{C}^S$, and parse $\mathcal{O}^H = \langle \mathtt{M}^O_{i_1}, \ldots, \mathtt{M}^O_{i_{n-|\mathbf{C}^S|}} \rangle$. For all $j \in [n - |\mathbf{C}^S|]$ add $(\hat{\mathtt{M}}_{i_j}, \mathtt{M}^O_{i_j})$ to Ψ_{in}.

15. Else if the message $(\texttt{write}, \texttt{invalid})$ appeared, send the following messages to the port $\texttt{OMix.infl}$:
    – For all $i \in \mathbf{C}^S$ send the message $(\texttt{input}, i, \mathsf{Dec}(\mathsf{sk}, \mathtt{C}_{0,j}))$;
    – send the message $\texttt{mixDone}$;

    Successively, receive the message $(\texttt{mixDone}, \texttt{invalid}, \mathcal{O})$, parse $\mathcal{O}$ as $\langle \mathtt{M}^O_1, \mathtt{M}^O_1, \ldots, \mathtt{M}^O_n \rangle$, for all $j \in [n] \setminus \mathbf{C}^S$ add $(\hat{\mathtt{M}}_j, \mathtt{M}^O_j)$ to Ψ_{in}.
16. Whenever a message $(\texttt{dec}, X, \kappa, \mathtt{M})$ appears either in the leakage port $\texttt{Dec.lk}$ or in the outport $\texttt{Dec.out}_{M_i}$ for any i substitute the message with $(\texttt{dec}, X, \kappa, \psi_{\mathsf{in}}(\psi_{\mathsf{hide}}(\mathtt{M})))$.

The final step is to prove that the last hybrid and the ideal world are indistinguishable.

Lemma 45. $\mathbf{H}_{10} \equiv (\mathcal{F}^{\texttt{OMix}} \circ \mathsf{S} \circ \mathcal{Z})$.

Proof. We start noticing that, by the changes introduced in steps $\mathbf{H}_0, \mathbf{H}_2, \mathbf{H}_4, \mathbf{H}_5, \mathbf{H}_6$, the aborting conditions of hybrid $\mathbf{H}_{10}$ and S are the same. Let $\texttt{Valid}$ be the event that $\mathbf{H}_{10}$ (resp. S) did not abort and the message $(\texttt{write}, \texttt{valid})$ appeared.

We show that conditioned on $\texttt{Valid}$, $\mathcal{Z}$ has the same view in both experiments. By the check in 10a and 10b of the simulator, all the honest simulated messages $\hat{\mathcal{M}}_H$ appear in $\mathcal{V}_m$ (or more formally in the projected set $\psi_{\mathsf{hide}}(\mathcal{V}_m)$) and moreover each of them appears once and only once. Therefore the simulator sends exactly $|\mathbf{C}^S|$ messages to the ideal functionality. So, summed to the inputs sent by the honest senders the ideal functionality received exactly n inputs.

In the decoding phase of the protocol both the simulator and the hybrid $\mathbf{H}_{10}$ compute a list of decryptions. In particular, we need to prove that the distributions of these lists as computed by the simulator and by the hybrid are indistinguishable. Both the simulator and the hybrid $\mathbf{H}_{10}$ compute the list as follow:

$$\langle \psi_{\mathsf{in}}(\psi_{\mathsf{hide}}(\mathsf{ADec}(\mathsf{sk}, \textstyle\prod_{l=1}^{m} \kappa_l, \mathtt{C}_{m,j}))) : j \in [n] \rangle.$$

Moreover:

1. Both the simulator and the hybrid compute the set Ψ_{hide} in the same way.
2. The hybrid $\mathbf{H}_{10}$ computes Ψ_{in} as the set of tuples $(\hat{\mathtt{M}}_j, \mathtt{M}_j)$ for $j \in [n] \setminus \mathbf{C}^S$, where $\mathtt{M}_j$ is the honest input message. The simulator instead computes Ψ_{in} by first looking at the list $\mathcal{O} \setminus \mathcal{O}^C$ (namely, the ordered list of the output messages minus the corrupted messages extracted by the simulator) and then by assigning to each simulated honest sender message $\hat{\mathtt{M}}_j$ a value from the list $\mathcal{O} \setminus \mathcal{O}^C$.

In particular, the two points above are already sufficient to show that, without considering the order, the two lists (as computed by the simulator or by hybrid) are the same. However, we still need to prove that the order does not allow to distinguish the two distributions.

The order of the corrupted inputs is equivalent in the two lists, so we focus on the order of the honest inputs.

Parse $[n] \setminus \mathbf{C}^S$ as $\{i_1, \ldots, i_{n-|\mathbf{C}^S|}\}$, and notice that by steps 13d and 10a of S we have that $\hat{\mathcal{M}}_H \subseteq \mathcal{V}_{h^*}$, and therefore:

– there exists a function π' such that $\mathtt{M}_{h^*,\pi'(i_j)} = \hat{\mathtt{M}}_{i_j}$ for all j and,

– there exists a set $\mathcal{M}' \subseteq \mathcal{M}_{\mathsf{hide}}$ such that $\psi_{\mathsf{hide}}(\mathcal{M}') = \hat{\mathcal{M}}_H$.

In particular, we can parse $\mathcal{M}' = \{\mathtt{H}_{\pi_{h^*}(\pi'(i_1))}, \ldots, \mathtt{H}_{\pi_{h^*}(\pi'(i_{n-|\mathbf{C}^S|}))}\}$. In $\mathbf{H}_{10}$ the following holds for all j:

$$\psi_{\mathsf{in}}(\psi_{\mathsf{hide}}(\mathtt{H}_{\pi_{h^*}(\pi'(i_j))})) = \psi_{\mathsf{in}}(\hat{\mathtt{M}}_{\pi'(i_j)}) = \mathtt{M}_{\pi'(i_j)},$$

while in the simulator the following holds for all j:

$$\psi_{\mathsf{in}}(\psi_{\mathsf{hide}}(\mathtt{H}_{\pi_{h^*}(\pi'(i_j))})) = \psi_{\mathsf{in}}(\hat{\mathtt{M}}_{\pi'(i_j)}) = \mathtt{M}^O_{\pi'(i_j)}.$$

However, notice that the view of $\mathcal{Z}$ is independent of the permutation π_{h^*}, restricted to the indices j such that $\mathtt{M}_{h^*-1,j} \neq \perp$, and in particular, since $\hat{\mathcal{M}}_H \subseteq \mathcal{V}_{h^*}$, the view of $\mathcal{Z}$ is independent of the permutation π_{h^*} restricted to the points $\{\pi'(i_j) : j \in [n] \setminus \mathbf{C}^S\}$. This implies that in both worlds each element $\mathtt{H}_{\pi_{h^*}(\pi'(i_j))}$ is assigned to a uniformly randomly selected honest message (without repetition, of course). Therefore the distributions of the decoded ciphertexts are equivalent in $\mathbf{H}_{10}$ and in the simulator.

Let `Invalid` be the event that the message $(\mathtt{write}, \mathtt{invalid})$ appeared. We show that the distribution of the decoded ciphertexts of S and $\mathbf{H}_{10}$ are exactly the same. We first analyze the inputs of the honest senders: conditioned on the event `Invalid`, the simulator assigns to $\hat{\mathtt{M}}_j$ the input of the honest sender $\mathtt{M}_j$ for any $j \in [n] \setminus \mathbf{C}^S$. The hybrid $\mathbf{H}_{10}$ does exactly the same thing. The inputs of the malicious senders given to the ideal functionality by S, conditioned on the event `Invalid`, are the decryptions of $\mathcal{L}_0$, so are the outputs. On the other hand, the hybrid $\mathbf{H}_{10}$, conditioned on the event `Invalid`, does decrypt the list $\mathcal{L}_0$ therefore obtaining the same messages.

Finally notice that the events `Valid`, `Invalid`, and the event that the simulator (resp. the hybrid) aborts partition the space of the possible outcomes. Therefore, we have proved all the possible cases.

By the triangular inequality and the lemmas above we get $(\Pi \circ \mathcal{F}^{\mathsf{Dec}} \circ \mathcal{F}^{\mathsf{NPRO}} \circ \mathrm{BB} \circ \mathcal{Z}) \approx_c (\mathcal{F}^{\mathtt{OMix}} \circ \mathsf{S} \circ \mathcal{Z})$.

Lemma 46. *Let $\mathcal{A} := (\mathcal{W}[\mathsf{S}] \circ \mathcal{W}[\mathcal{F}^{\mathsf{Dec}}] \circ \mathcal{F}^{\mathsf{NPRO}} \circ \mathrm{BB})$, for all environment $\mathcal{Z} \in Env$, $(\mathcal{Z} \circ \mathcal{A} \circ \mathcal{P}_A[\mathsf{Audit}]) \equiv (\mathcal{Z} \circ \mathcal{A} \circ \mathcal{P}_A[\mathsf{Audit}^*_{\mathtt{OMix}}])$.*

Proof. Suppose the two hybrids diverge. Then it must be that the environment $\mathcal{Z}$ sent an input $(b^*, \mathcal{O}^*)$ to $\mathcal{P}_A$ such that Audit and $\mathsf{Audit}^*_{\mathtt{OMix}}$ answer differently.

Suppose that Audit returns false while $\mathsf{Audit}^*_{\mathtt{OMix}}$ returns true. If $\mathsf{Audit}^*_{\mathtt{OMix}}$ returns true, it means that the simulator sent `mixDone` to the influence port of ideal functionality $\mathcal{F}^{\mathtt{OMix}}$ and the message $(b^*, \mathcal{O}^*)$ appears in $\mathcal{D}^I$. Suppose that Audit return false because $\mathsf{flg}_{\mathtt{valid}} \neq b^*$, but notice that if $\mathsf{flg}_{\mathtt{valid}}$ appears the BB the simulator accordingly to its value sends the message `invalidate` to the ideal functionality. In particular when $\mathsf{flg}_{\mathtt{valid}} = \mathtt{valid}$ then the simulator does not send the `invalidate` message and therefore b^* cannot be `invalid` and when $\mathsf{flg}_{\mathtt{valid}} = \mathtt{invalid}$ then the simulator does send the message `invalidate` and therefore b^* cannot be `valid`. This implies that $b^* = \mathsf{flg}_{\mathtt{valid}}$.

If $b^* = \mathtt{valid}$ then the checks (1) and (2) of the Audit algorithm hold true, moreover, $\perp \notin \mathcal{V}_m$ so all the ciphertexts in the list $\mathcal{L}_m$ decrypt correctly. The latter implies that the simulator in step (15) when receives messages of the form $(\mathtt{dec}, \mathtt{C}^*_j, \kappa^*, \mathtt{M})$ it substitute it with $(\mathtt{dec}, \mathtt{C}^*_j, \kappa^*, \Psi_{\mathsf{in}}(\Psi_{\mathsf{hide}}(\mathtt{M})))$ where the maps are programmed to outputs exactly what is $\mathcal{O}^*$, therefore also check (3) holds true.

If $b^* = \mathtt{invalid}$ then either the adversary sent `invalidate` so either (1) or (2) are false or $\perp \in \mathcal{V}_m$. In step (15) when receives messages of the form $(\mathtt{dec}, \mathtt{C}^*_j, \kappa^*, \mathtt{M})$ it substitute it with $(\mathtt{dec}, \mathtt{C}^*_j, \kappa^*, \Psi_{\mathsf{in}}(\Psi_{\mathsf{hide}}(\mathtt{M})))$ where the map Ψ_{in} is programmed to outputs all the input messages,

therefore, if (1) and (2) are true it must be that check (3) holds false as otherwise $\mathcal{O}^*$ is not the same as the one output by the ideal functionality.

The other case – Audit returns true but $\mathsf{Audit}^*_{\mathtt{Mix}}$ returns false – simply cannot happen by definition of S (as the simulator does use the ideal functionality to produce its output).

This concludes the proof of the Theorem.

6 Verifiable Mixing from Optimistic Mixing

We propose a construction of a verifiable mixing scheme obtained by combining an optimistic mixing and a verifiable mixing scheme. The main feature of the mixing scheme that we obtain is that, in case all the participants behave correctly, it runs (approximately) as efficiently as the underlying optimistic mixing scheme. At the same time, it achieves the same security property of a verifiable mixing. More specifically, our construction uses a specific class of VMix schemes, that we call *structured* VMix, and that satisfy specific structural properties. We define structured VMix in the next section and then present our compiler in Section 6.2.

6.1 Structured Verifiable Mixing Scheme

Informally speaking, a Structured Verifiable Mixing (sVMix) scheme is a verifiable mixing scheme where: every sender submits its input by creating an encoding of it (e.g., using a public algorithm Encode), and the output is a sorted list of the decoded inputs. Also, the scheme leaves the possibility releasing the secret key if mixing did not happen. This behavior and capabilities are quite natural for mixing protocols and indeed capture many existing schemes. In Sec. 6.3, we indeed show how to instantiate it using the UC-secure mixing protocol of [51]. Furthermore, for the sake of our compiler we need a structured VMix that satisfies a security property weaker and simpler than the one of a standard VMix. Namely, inputs of the ideal functionality are submitted (in encoded form) by one single party all at once.

More formally, we define a Structured Verifiable Mixing (sVMix) scheme as a tuple $\mathcal{SVM} = (\mathcal{ES}, \Pi, \mathsf{Audit})$ where $\mathcal{ES}$ is a quintuple of PPT algorithms $(\mathsf{Init}, \mathsf{Encode}, \mathsf{Decode}, \mathsf{Encode}^X, \mathsf{Decode}^X)$ representing an encoding scheme as defined below, Π is a multi-party protocol, and Audit is a PPT algorithm.

For the encoding scheme $\mathcal{ES}$ we consider a specific kind of schemes in which the Encode algorithm outputs a pair $Z = (X, Y)$ and, very informally, restricting ciphertexts to the X component yields an correct encryption scheme (notably, we do not require any kind of security) while full ciphertexts yield an IND-CCA2-secure one.

This is essentially a specialization of the notion of *augmented cryptostystem* that was introduced of Wikström [52] as a tool for submitting inputs to mix-net protocols, and captures several IND-CCA2 encryption schemes, such as Cramer-Shoup (where Y is a well specified component of the ciphertext), or ones following the Naor-Yung paradigm (where Y is essentially a proof of knowledge of the plaintext encrypted in X). More formally:

Definition 12 (Encoding Scheme). *An encoding scheme* $\mathcal{ES} = (\mathsf{Init}, \mathsf{Encode}, \mathsf{Decode}, \mathsf{Encode}^X, \mathsf{Decode}^X)$ *is one where* $(\mathsf{Init}, \mathsf{Encode}, \mathsf{Decode})$ *is a correct encryption scheme, and the following extra properties hold:*

- $\mathsf{Encode}(\mathsf{pub}, \mathtt{M})$ *outputs an encoding* $Z = (X, Y)$*;*
- Encode^X *is the algorithm that runs* Encode *and returns* X *only;*
- *The algorithm* $\mathsf{Decode}(\mathsf{sec}, Z)$ *works by computing* $\mathsf{Decode}^X(\mathsf{sec}, \mathsf{Strip}(\mathsf{sec}, Z))$ *where* Strip[8] *is an algorithm that on input* $Z = (X, Y)$ *outputs* X *or* $\perp$*.*

[8] While here we let Strip work with the secret key sec, the algorithm Strip could also work only with public parameter

Definition 13 (Submission security). *An encoding scheme $\mathcal{ES}$ as above is submission-secure if (i) the tuple* $(\mathsf{Init}, \mathsf{Encode}, \mathsf{Decode})$ *is IND-CCA2 secure, and (ii) the tuple* $(\mathsf{Init}, \mathsf{Encode}^X, \mathsf{Decode}^X)$ *is a correct encryption scheme*[9].

To model security of sVMix, in Fig. 7 we describe an ideal functionality $\mathcal{F}^{\mathtt{sMix}}[\mathcal{ES}]$, parametrized by the encoding scheme $\mathcal{ES}$. At high level, $\mathcal{F}^{\mathtt{sMix}}[\mathcal{ES}]$ works as follows. In the **Init** phase, it generates the keys $(\mathsf{pub}, \mathsf{sec})$ of the encoding scheme. For **input submission**, it waits a message from the first mixer $\mathcal{P}_{M_1}$ containing a list of encoded inputs $\{Z_j\}_{j\in[n]}$. In the **mixing** phase, it waits a confirmation from every mixer and then outputs the sorted list $\mathsf{Sort}(\langle \mathsf{Decode}(\mathsf{sec}, Z_j)\rangle_{j\in[n]})$. Finally, the functionality has a command **Unveil** that, when all mixers agree, release the secret parameter sec.

We are ready to define the security property of a structured mixing protocol:

Definition 14. *An sVMix scheme $\mathcal{SVM} = (\mathcal{ES}, \Pi, \mathsf{Audit})$ is $\mathcal{R}$-*secure *iff: (1) $\mathcal{SVM}$ is a secure auditable protocol for $\mathcal{F}^{\mathtt{sMix}}[\mathcal{ES}]$ with resource $\mathcal{R}$ for environments in Env^{static}; (2) $\mathcal{ES}$ is submission-secure.*

Functionality $\mathcal{F}^{\mathtt{sMix}}[\mathcal{ES}]$:

The functionality is implicitly parametrized by the security parameter 1^λ.

Init: At first activation read from $\mathtt{sMix.infl}$ the sets of corrupted players $\mathbf{C}^M$, compute $(\mathsf{pub}, \mathsf{sec}) \leftarrow \mathsf{Init}(1^\lambda)$, set $\mathcal{I}^S \leftarrow \emptyset$, $\mathcal{I}^M \leftarrow \emptyset$, $\mathsf{flg}_{\mathtt{inpDone}} \leftarrow \mathsf{false}$, and $\mathsf{flg}_{\mathtt{mixDone}} \leftarrow \mathsf{false}$. Send the message $(\mathtt{pk}, \mathsf{pub})$ on outport $\mathtt{sMix.lk}$ and return.

Input Submission: On message $(\mathtt{input}, \langle Z_j\rangle_{j\in[n]})$ on the inport $\mathtt{sMix.in}_{M_1}$ (or from $\mathtt{sMix.infl}$ if $M_1 \in \mathbf{C}^M$) set $\mathcal{I}^Z \leftarrow \langle Z_j\rangle_{j\in[n]}$; Send $(\mathtt{input}, \mathcal{I}^Z)$ to the leakage port and set $\mathsf{flg}_{\mathtt{input}}$ to true.

Mix: On message $\mathtt{mix}$ on the inport $\mathtt{sMix.in}_{M_i}$ add the index i in the set of the the mixers' inputs $\mathcal{I}^M$ and send $(\mathtt{mix}, i)$ to the leakage port; Moreover, if $|\mathcal{I}^M| = m$ then set $\mathsf{flg}_{\mathtt{mixDone}}$ to true and set $\mathcal{O} \leftarrow Sort(\langle \mathsf{Decode}(\mathsf{sec}, Z_j)\rangle_{j\in[n]})$

Unveil: On message $\mathtt{unveil}$ on the protocol input ports $\mathtt{in}_{M_i}$ store the index i in the set $\mathcal{I}$, if $|\mathcal{I}| = m$ then send the message $(\mathtt{unveil}, \mathsf{sec})$ to the leakage port;

Delivery:

- On message $(\mathtt{pk}, i)$ on the influence port, send the message $(\mathtt{pk}, \mathsf{pub})$ to the port $\mathtt{out}_{M_i}$;
- On message $(\mathtt{mixDone}, i)$ on the inport $\mathtt{sMix.infl}$, if $\mathsf{flg}_{\mathtt{mixDone}}$ is true then send the message $(\mathtt{mixDone}, \mathcal{O})$ on the outport $\mathtt{sMix.out}_{M_i}$.
- On message $(\mathtt{unveil}, i)$ on the influence port if $|\mathcal{I}| = n$ send the message $(\mathtt{unveil}, \mathsf{sec})$ to the port $\mathtt{out}_{M_i}$;

Fig. 7: Ideal Functionality for *"Structured"* Mixing.

6.2 Construction

Let $\mathcal{PKE}^C = (\mathsf{KGen}^C, \mathsf{Enc}^C, \mathsf{Dec}^C)$ be an IND-CPA secure encryption scheme, and let $\mathcal{F}^{\mathtt{DKG}}$ be the ideal key generation functionality parametrized for KGen^C (see Fig. 8).

Given a structured VMix scheme $\mathcal{SVM} = (\mathcal{ES}, \Pi, \mathsf{Audit})$, an OMix scheme $\mathcal{OM} = (\Pi', \mathsf{Audit}')$, we construct a VMix scheme $\mathcal{VM} = (\Pi'', \mathsf{Audit}'')$ as follows.

The protocol. The protocol Π'' with resources $\mathcal{F}^{\mathtt{sMix}}[\mathcal{ES}]$, $\mathcal{F}^{\mathtt{OMix}}$ and $\mathcal{F}^{\mathtt{DKG}}[\mathsf{KGen}^C]$:

[9] Namely, that for any pair $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{Init}$, any message $\mathtt{M}$, the equation $\mathsf{Decode}(\mathsf{sec}, \mathsf{Encode}(\mathsf{pub}, \mathtt{M})) = \mathtt{M}$ holds true.

1. Encode and Input Submission. The sender $\mathcal{P}_{S_j}$ on input $\mathtt{M}_j$, reads from the input protocol port of $\mathtt{sMix}$ the message $(\mathsf{pk}, \mathsf{pub})$, computes $Z_j := (X_j, Y_j) \leftarrow \mathsf{Encode}(\mathsf{pub}, \mathtt{M}_j)$, compute $\Psi_j \leftarrow \mathsf{Enc}^C(\mathsf{pk}, Y_j)$, send the message $(\mathtt{write}, (j, \Psi_j))$ to the outport $\mathtt{BB.in}_{S_j}$, and sends the message $(\mathtt{input}, X_j)$ to the outport $\mathtt{OMix.in}_{S_j}$.
2. Mix. The mixer $\mathcal{P}_{M_i}$ reads the message $(\mathtt{mixDone}, b, \mathcal{O}')$ from the inport $\mathtt{OMix.out}_{M_i}$, parses $\mathcal{O}' = \langle X'_j \rangle_{j \in [n]}$, and sends the message $(\mathtt{write}, (b, \mathcal{O}'))$ to the port $\mathtt{BB.in}_{M_i}$;
   Valid Branch: (If $b = \mathtt{valid}$) Send the message $\mathtt{unveil}$ to the outport $\mathtt{sMix.in}_{M_i}$ and return; at next activation read the messages $(\mathtt{unveil}, \mathsf{sec})$, send the message $(\mathtt{write}, \mathsf{sec})$ to the input port of $\mathtt{BB}$, and compute $\mathtt{M}_j \leftarrow \mathsf{Decode}^X(\mathsf{sec}, X'_j)$ for $j \in [n]$ and output $(\mathtt{mixDone}, \mathcal{O}'')$ where $\mathcal{O}'' \leftarrow \mathsf{Sort}(\langle \mathtt{M}_j \rangle_{j \in [n]})$.
   Invalid Branch: (If $b = \mathtt{invalid}$)
   (a) Send the message $\mathtt{unveil}$ to the port $\mathtt{DKG.in}_{M_i}$ and the messages $(\mathtt{read}, j)$ for $j \in [n]$ to the port $\mathtt{BB.in}_{M_i}$ and return; At next activation read the message $(\mathtt{unveil}, \mathsf{sk})$ from the port $\mathtt{DKG.out}_{M_i}$ and read $\{(\mathtt{read}, (j, \Psi_j))\}_{j \in [n]}$ from port $\mathtt{BB.out}_{M_i}$ and compute the $Y'_j \leftarrow \mathsf{Dec}^C(\mathsf{sk}, \Psi_j)$ and set $Z'_j \leftarrow (X'_j, Y'_j)$ for all $j \in [n]$,
   (b) If $i = 1$ send the message $(\mathtt{input}, \langle Z'_j \rangle_{j \in [n]})$ and $\mathtt{mix}$ to the outport $\mathtt{sMix.in}_{M_1}$;
   (c) If $i \neq 1$ read the message $(\mathtt{input}, \mathcal{I}^Z)$ from the port $\mathtt{sMix.out}_{M_i}$; if $\mathcal{I}^Z = \langle Z'_j \rangle_{j \in [n]}$ send the message $\mathtt{mix}$ to the outport $\mathtt{sMix.in}_{M_1}$;
   (d) Read the message $(\mathtt{mixDone}, \mathcal{O})$ from $\mathtt{sMix.out}_{M_i}$ and output $(\mathtt{mixDone}, \mathcal{O}'')$ where $\mathcal{O}'' \leftarrow \mathcal{O}$;
3. All parties send the message $(\mathtt{write}, \mathtt{endProtocol})$ to the relative protocol input port of BB.

The Audit Algorithm. The algorithm Audit takes as input $(\tau, \mathcal{O})$, and works as follows:

1. Parse τ as $(\mathcal{D}^R, \mathcal{D}^I)$, and initialize $\mathcal{O}'' \leftarrow \emptyset$;
2. Check if for all $i \in [m]$ there exists an entry $(*, M_i, *, \mathcal{O}')$ in $\mathcal{D}^R$, otherwise return false;
3. If for all $i \in [m]$ there exists an entry $(*, M_i, (\mathtt{valid}, \mathcal{O}'))$ in $\mathcal{D}^R$:
   (a) Check if $\mathsf{Audit}^*_{\mathtt{OMix}}(\mathcal{D}^I, (\mathtt{mixDone}, \mathtt{valid}, \mathcal{O}')) = 1$, otherwise return false;
   (b) Check that $\mathsf{Audit}^*_{\mathtt{sMix}}(\mathcal{D}^I, (\mathtt{unveil}, \mathsf{sec})) = 1$ where $(*, *, \mathsf{sec}) \in \mathcal{D}^R$, for all $X \in \mathcal{O}'$, check if there exists $\mathtt{M} \in \mathcal{O}$ such that $\mathsf{Decode}^X(\mathsf{sec}, X) = \mathtt{M}$, and if so add the message $\mathtt{M}$ in the list $\mathcal{O}''$;
   (c) Return true if $\mathsf{Sort}(\mathcal{O}'') = \mathcal{O}$, false otherwise.
4. Else return true if all the following checks are satisfied, and false otherwise: $\mathsf{Audit}^*_{\mathtt{OMix}}(\mathcal{D}^I, (\mathtt{mixDone}, \mathtt{invalid}, \mathcal{O}')) = 1$, $\mathsf{Audit}^*_{\mathtt{sMix}}(\mathcal{D}^I, (\mathtt{mixDone}, \mathcal{O})) = 1$, and $\mathsf{Audit}^*_{\mathtt{DKG}}(\mathcal{D}^I, (\mathtt{unveil}, \mathsf{sk}))$ 1.

Functionality $\mathcal{F}^{\mathtt{DKG}}[\mathsf{KGen}]$:

The functionality has m parties with indexes $\{M_i : i \in [m]\}$. The functionality is implicitly parametrized by the security parameter 1^λ.

Init: At first activation sample $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$ and set the set $\mathcal{I}$ to empty;
Unveil: On message $\mathtt{unveil}$ on the protocol input ports $\mathtt{in}_{M_i}$ store the index i in the set $\mathcal{I}$, if $|\mathcal{I}| = m$ then send the message $(\mathtt{unveil}, \mathsf{sk})$ to the leakage port;
Delivery: On message $(\mathtt{pubkey}, i)$ on the influence port send the message $(\mathtt{pubkey}, \mathsf{pk})$ to the port $\mathtt{out}_{M_i}$; On message $(\mathtt{unveil}, i)$ on the influence port if $|\mathcal{I}| = n$ send the message $(\mathtt{unveil}, \mathsf{sk})$ to the port $\mathtt{out}_{M_i}$;

Fig. 8: Ideal functionality for Distributed Key Generation

In what follows we prove that the protocol $\mathcal{VM}$ above is a secure VMix. The interesting part of the construction above is its computational complexity. Informally, whenever the adversary

$\mathcal{Z}$ does not invalidate the optimistic execution, the execution time of the protocol depends only on the complexity of the OMix protocol. We formalize this intuitive property in the statement of the theorem below:

Theorem 7. *Let $\mathcal{PKE}^C = (\mathsf{KGen}^C, \mathsf{Enc}^C, \mathsf{Dec}^C)$ be an IND-CPA secure encryption scheme, $\mathcal{SVM} = (\mathcal{ES}, \Pi', \mathsf{Audit}')$ be a $\mathcal{F}$-secure structured VMix for any $\mathcal{F}$, and $\mathcal{OM} = ()$ The following two statements are both true: (1) $\mathcal{VM}$ is a $(\mathcal{F}^{\mathit{sMix}}[\mathcal{ES}], \mathcal{F}^{\mathit{OMix}}, \mathcal{F}^{\mathsf{DKG}}[\mathsf{KGen}^C])$-secure verifiable mixing scheme, (2) for any execution of the protocol, if the message* `invalid` *does not appear in the port* `OMix.infl` *then no message of the form* $(\texttt{input}, *)$ *appears in the outports of* `sMix`*.*

Proof. We prove the theorem with a series of hybrids. **Hybrid 1.** Let $\mathbf{H}_1(\lambda)$ be the experiment that runs the interactive agent $(\mathcal{Z} \circ \Pi'' \circ \mathcal{F}^{\mathtt{Mix}}[\mathcal{ES}] \circ \mathcal{F}^{\mathtt{OMix}} \circ \mathcal{F}^{\mathsf{DKG}} \circ \mathrm{BB})$ on input the security parameter. The hybrid sees the internal views of all the agents and keeps track of all the messages sent. Moreover the hybrid does the following modifications:

- When the message $(\texttt{input}, j, X_j)$ appears in the port of $\mathtt{OMix.in}_{S_j}$:
  - If $j \in \mathbf{C}^S$ then register the tuple $(j, X_j, \perp)$ in the database of the encoded inputs
  - If $j \in [n] \setminus \mathbf{C}^S$, compute $\hat{Z}_j = (\hat{X}_j, \hat{Y}_j) \leftarrow \mathsf{Encode}(\mathsf{pub}, 0)$, store the tuple $(j, X_j, \hat{X}_j)$ in the database of the encoded inputs, compute $\Psi'_j \leftarrow \mathsf{Enc}^C(\mathsf{pk}, Y_j)$, and send the message $(\texttt{write}, (j, \Psi'_j))$ to the bulletin board.
- When the message $(\texttt{mixDone}, b, \mathcal{O}')$ appears from the protocol ports of `OMix`, replace it with $(\texttt{mixDone}, b, \hat{\mathcal{O}}')$ where $\hat{\mathcal{O}}'$ is computed as follows:
  - If $b = \texttt{valid}$ let $\hat{\mathcal{O}}' \leftarrow \mathsf{Sort}(\langle X_j \rangle_{j \in [n]})$, where $(j, X_j, *)$ appears in the database of the encoded inputs.
  - If $b = \texttt{invalid}$ let $\hat{\mathcal{O}}' \leftarrow \langle X'_j \rangle_{j \in [n]}$ where X'_j:

$$X'_j \leftarrow \begin{cases} X_j \; j \in \mathbf{C}^S \\ \hat{X}_j \quad \text{else} \end{cases}$$

Lemma 47. $(\mathcal{Z} \circ \Pi'' \circ \mathcal{F}^{\mathit{sMix}}[\mathcal{ES}]\mathcal{F}^{\mathit{OMix}} \circ \mathcal{F}^{\mathsf{DKG}} \circ \mathrm{BB}) \approx_c \mathbf{H}_1$.

Proof. To prove the lemma, we define another sequence of hybrid experiments $\mathbf{H}_{1,i}$ for $i \in [n]$, where $\mathbf{H}_{1,i}$ is the same as $\mathbf{H}_1$ except that:

- When both the messages $(\texttt{input}, j, X_j)$ and $(\texttt{write}, (j, \Psi_j))$ appear in the outports of the party $\mathcal{P}_{S_j}$:
  - If $j \in \mathbf{C}^S$ <u>or $j \geq i$</u>, then register the tuple $(j, X_j, \perp)$ in the database of the encoded inputs,
  - If $j \in [n] \setminus \mathbf{C}^S$ <u>and $j < i$</u> compute $\hat{Z}_j := (\hat{X}_j, \hat{Y}_j) \leftarrow \mathsf{Encode}(\mathsf{pub}, 0)$ and store the tuple $(j, X_j, \hat{X}_j)$ in the database of the encoded inputs, compute $\Psi'_j \leftarrow \mathsf{Enc}^C(\mathsf{pk}, Y_j)$, and send the message $(\texttt{write}, (j, \Psi'_j))$ to the bulletin board.
- When the message $(\texttt{mixDone}, b, \mathcal{O}')$ appears from the protocol ports of `OMix` do as follow:
  - If $b = \texttt{valid}$ let $\hat{\mathcal{O}}' \leftarrow \mathsf{Sort}(\langle X_j \rangle_{j \in [n]})$ where $(j, X_j, *)$ appears in the database of the encoded inputs.
  - If $b = \texttt{invalid}$ let $\hat{\mathcal{O}}' \leftarrow \langle A_j \rangle_{j \in [n]}$ where:

$$A_j \leftarrow \begin{cases} X_j \text{ if } j \in \mathbf{C}^S \text{ or } j \geq i \\ \hat{X}_j \text{ if } j \in [n] \setminus \mathbf{C}^S \text{ and } \underline{j < i} \end{cases}$$

Clearly, $\mathbf{H}_{1,i}$ executes exactly the same as the real experiment, while $\mathbf{H}_{1,n+1} \equiv \mathbf{H}_1$.

Without loss of generality we consider an environment $\mathcal{Z}$ that outputs only a single bit. We rely on the weak submission security of $\mathcal{ES}$ and on the IND-CPA security of the encryption scheme $\mathcal{PKE}$. Specifically, consider two classes of environments: the class Env_1 such that, for all $\mathcal{Z} \in Env_1$, $\mathcal{Z}$ does static and active corruption of the players and the message `invalidate` is sent with negligible probability and the class Env_2 such that for all $\mathcal{Z} \in Env_2$ the environment $\mathcal{Z}$ does static corruption of the players and the message `invalidate` is sent with noticeable probability.
Notice that $Env_2 \cup Env_1 = Env^{\mathsf{static}}$.

Claim 21 *For all $\mathcal{Z} \in Env_1$ and for all $i \in [n]$, $\mathbf{H}_{1,i} \approx_c \mathbf{H}_{1,i+1}$.*

Proof. In this case the message `invalidate` is sent only with negligible probability by $\mathcal{Z}$. By inspection, we notice that conditioned on the event that `invalidate` is not sent, the encoded messages $\langle \hat{X}_i \rangle_{i \in [n] \setminus \mathbf{C}^S}$ do not appear in the view of the environment. However, notice that while $\mathbf{H}_{1,i}$ produces Ψ'_i as an encryption of Y_i, $\mathbf{H}_{1,i+1}$ produces Ψ'_i as an encryption of $\hat{Y}_i$. Moreover, in this conditional space the secret key sk for the encryption scheme $\mathcal{PKE}^C$ is never unveiled. Therefore, we can easily reduce to the IND-CPA security of $\mathcal{PKE}^C$. So in this conditional space the two views are computationally close. This is enough to prove that, even without conditioning on this event, the views are computationally close, because the event happens only with negligible probability.

Claim 22 *For all $\mathcal{Z} \in Env_2$ and for all $i \in [n]$, $\mathbf{H}_{1,i} \approx_c \mathbf{H}_{1,i+1}$.*

Proof. In this case the message `invalidate` is sent with noticeable probability $1/p(\lambda)$ by $\mathcal{Z}$. We show how to reduce to the submission security of $\mathcal{ES}$ (specifically, to the IND-CCA2 security of $(\mathsf{Init}, \mathsf{Encode}, \mathsf{Decode})$). We consider a reduction that simulates correctly only the invalid branch and aborts otherwise. The reduction is almost straightforward to the IND-CCA2 experiment. More details follow:

Adversary $\mathsf{B}(\mathsf{pub})$:

1. Run the same code of $\mathbf{H}_{1,i}$ but the agent $\mathcal{F}^{\mathtt{sMix}}[\mathcal{ES}]$, instead of sampling from $\mathsf{Init}(1^\lambda)$, sets the public key as pub; (In particular, it does not know the relative secret key.)
2. When a honest sender $\mathcal{P}_{S_j}$, with $j \in [n] \setminus \mathbf{C}^S$, is activated and receives its input $\mathtt{M}_j$, store $\mathtt{M}_j$, and if $j \neq i$ keep on simulating the hybrid $\mathbf{H}_{1,i}$ by encrypting either $\mathtt{M}_j$ or 0; else if $j = i$ send to the challenger the tuple $(\mathtt{M}_i, 0)$; get a ciphertext $(\hat{X}_i, \hat{Y}_i)$;
3. When a message $(\mathtt{input}, j, X_j)$ appears in a port $\mathtt{OMix.infl}_{S_j}$ and a message $(\mathtt{write}, (j, \Psi_j)$ appears in the port $\mathtt{BB.in}_{S_j}$ with $j \in \mathbf{C}^S$, first decrypt $Y_j \leftarrow \mathsf{Dec}^C(\mathsf{sk}, \Psi_j)$ the ciphertext and then query (X_j, Y_j) to the decryption oracle, obtain $\mathtt{M}_j$ and store it;
4. If the message `mixDone` is sent by the environment, all the senders have sent their inputs to the functionality $\mathcal{F}^{\mathtt{OMix}}$ but the message `invalidate` has not been sent then abort; (Let `Abort` be such event.)
5. Simulate the ideal functionality $\mathcal{F}^{\mathtt{sMix}}[\mathcal{ES}]$ computing $\mathcal{O} \leftarrow \mathsf{Sort}(\langle \mathtt{M}_j \rangle_{j \in [n]})$.
6. At the end of the experiment output whatever $\mathcal{Z}$ outputs.

When the challenge bit b^* is equal to 0 then the challenger encrypts the message $\mathtt{M}_i$, and thus B perfectly simulates the view of $\mathcal{Z}$ in $\mathbf{H}_{1,i}$ conditioned on the event $(\neg\mathtt{Abort})$. On the other hand, when $b^* = 1$ the message encrypted is 0, and therefore B perfectly simulates the view of $\mathcal{Z}$ in $\mathbf{H}_{1,i+1}$, again conditioned on the event $(\neg\mathtt{Abort})$. Let $\epsilon := |\Pr[\mathbf{H}_{1,i} = 1] - \Pr[\mathbf{H}_{1,i+1} = 1]|$, where the hybrids run with environment $\mathcal{Z} \in Env^2$. By standard calculation we have that the advantage of B in the IND-CCA2 experiment is at least $1/2p(\lambda)\epsilon$.

We define the simulator.

Simulator S:
The simulator internally runs the open agent system $(\Pi'' \circ \mathcal{F}^{\mathtt{sMix}}[\mathcal{ES}] \circ \mathcal{F}^{\mathtt{OMix}} \circ \mathcal{F}^{\mathsf{DKG}})$ on input the security parameter. The simulator sees the internal views of all the agents and keeps track of all the messages sent, and also interacts externally with the agents $\mathcal{Z}$ and $\mathcal{F}^{\mathtt{Mix}}$ and forwards all the messages from/to $\mathcal{Z}$. Moreover, the simulator does the following modifications:

- When the message $(\mathtt{input}, j)$ appears in the leakage port of $\mathcal{F}^{\mathtt{sMix}}$ (the port that connects the simulator with the ideal functionality) send the input 0 to the party $\mathcal{P}_{S_j}$ of the internally simulated protocol Π'', the party computes $(\hat{X}_j, \hat{Y}_j) \leftarrow \mathsf{Encode}(\mathsf{pub}, 0)$ and $(\hat{\Psi}_j \leftarrow \mathsf{Enc}^C(\mathsf{pk}, \hat{Y}_j)$ and sends the relative messages. Store the tuple $(i, \hat{X}_j, \hat{Y}_j, \perp)$ in the database of the inputs.
- When the message X_j appears in the influence port of $\mathcal{F}^{\mathtt{OMix}}$ and the message $(\mathtt{write}, \Psi_j)$ appears in the influence port $\mathtt{OMix.infl}_{S_j}$ (both ports connect the simulator with the environment) then compute $Y_j \leftarrow \mathsf{Dec}^C(\mathsf{sk}, \Psi_j)$ and register the tuple $(j, X_j, Y_j, \perp)$ in the database of the inputs.
- If the message `invalidate` appears in the influence port of $\mathcal{F}^{\mathtt{OMix}}$ set the bit b to `invalid`.
- When the message `mixDone` appears in the influence port of $\mathcal{F}^{\mathtt{OMix}}$ (the message comes from the environment),
  - if $b =$ `valid` for every entry $(j, X_j, Y_j, \perp)$ where $j \in \mathbf{C}^S$ in the database of the inputs compute $\mathtt{M}_j \leftarrow \mathsf{Decode}^X(\mathsf{sec}, X_j))$ and update the entry to $(i, X_j, Y_j, \mathtt{M}_j)$ and send the message $(\mathtt{input}, j, \mathtt{M}_j)$ to the influence port of the ideal functionality $\mathcal{F}^{\mathtt{sMix}}$.
  - if $b =$ `invalid` for every entry $(j, X_j, Y_j, \perp)$ where $j \in \mathbf{C}^S$ in the database of the inputs compute $\mathtt{M}_j \leftarrow \mathsf{Decode}(\mathsf{sec}, (X_j, Y_j))$ and update the entry to $(i, X_j, Y_j, \mathtt{M}_j)$ and send the message $(\mathtt{input}, j, \mathtt{M}_j)$ to the influence port of the ideal functionality $\mathcal{F}^{\mathtt{sMix}}$.

  Next, send `mixDone` to the influence port of the ideal functionality $\mathcal{F}^{\mathtt{Mix}}$, and receive the message $(\mathtt{mixDone}, \mathcal{O}'')$. Proceed as follows:
  - If $b =$ `valid` let $\langle \mathtt{M}_1^O, \ldots, \mathtt{M}^O_{n-|\mathbf{C}^S|} \rangle$ be the list of honest inputs computed as $\mathcal{O}'' \backslash \langle \mathsf{Decode}(\mathsf{sec}, X_j) \rangle_{j \in \mathbf{C}^S}$, let $\hat{\mathcal{O}}' \leftarrow \mathsf{Sort}(\ \langle X_j \rangle_{j \in \mathbf{C}^S}, \langle \mathsf{Encode}^X(\mathsf{pub}, \mathtt{M}_j^O) \rangle_{j \in [n] \backslash \mathbf{C}^S}\ )$, where $(j, X_j, Y_j, \mathtt{M}_j)$ appears in the database of the encoded inputs.
  - If $b =$ `invalid` let $\hat{\mathcal{O}}' \leftarrow \langle X'_j \rangle_{j \in [n]}$ where X'_j:

$$X'_j \leftarrow \begin{cases} X_j \ j \in \mathbf{C}^S \\ \hat{X}_j \quad \text{else} \end{cases}$$

  Continue and trivially simulate the ideal functionality $\mathcal{F}^{\mathtt{sMix}}[\mathcal{ES}]$ using $\hat{\mathcal{O}}'$ and the functionality $\mathcal{F}^{\mathsf{DKG}}$.

Lemma 48. $\mathbf{H}_1 \equiv (\mathcal{Z} \circ \mathsf{S} \circ \mathcal{F}^{Mix})$.

Proof. By simple inspection, the view of the $\mathcal{Z}$ when the message `invalidate` is sent is the same in $\mathbf{H}_1$ and in the simulated world. On the other hand, consider the two views conditioned on the event that the message `invalidate` is not sent. In this case, the simulator gets the output messages from the ideal functionality and then creates the encoding $E^{\mathsf{S}} \leftarrow \langle X_j \leftarrow \mathsf{Encode}^X(\mathsf{pub}, \mathtt{M}_j^O) \rangle_{j \in [n] \backslash \mathbf{C}^S}$. On the other hand the hybrid receives the encoded messages $E^{\mathbf{H}} \leftarrow \langle X_j : \ (X_j, Y_j) \leftarrow \mathsf{Encode}(\mathsf{pub}, \mathtt{M}_j) \rangle_{j \in [n] \backslash \mathbf{C}^S}$. We notice that, if we consider the order, then the two distributions are different. However, the environment receives them sorted, as the optimistic mixing functionality executes the valid branch. In this case the two distributions are equivalent.

Notice that the two lemmas above implies the first condition of Def. 6 (secure auditable protocols). We now prove the second condition.

Lemma 49. *Let* $\mathcal{A} := (\mathcal{W}[\mathsf{S}] \circ \mathcal{W}[\mathcal{F}^{Mix}] \circ \mathrm{BB})$*, for all environment* $\mathcal{Z} \in Env$*,* $(\mathcal{Z} \circ \mathcal{A} \circ \mathcal{P}_A[\mathsf{Audit}]) \equiv (\mathcal{Z} \circ \mathcal{A} \circ \mathcal{P}_A[\mathsf{Audit}^*_{\mathtt{VMix}}])$.

Proof. Suppose the two hybrids diverge. Then it must be that the environment $\mathcal{Z}$ sent an input $\mathcal{O}^*$ to $\mathcal{P}_A$ such that Audit and $\mathsf{Audit}^*_{\mathtt{Mix}}$ answer differently.

Suppose that Audit returns false while $\mathsf{Audit}^*_{\mathtt{Mix}}$ returns true. If $\mathsf{Audit}^*_{\mathtt{Mix}}$ returns true, it means that the simulator sent `mixDone` to the ideal functionality, and the simulator, by definition, sends the messages $(\mathtt{write}, M_i, b, \mathcal{O}')$ to the BB for all $i \in [m]$. Hence, the algorithm Audit does not return false as result of the check in step 2. Furthermore, if `mixDone` was sent by S to the ideal functionality $\mathcal{F}^{\mathtt{Mix}}$, by construction of S it must be that a message $(\mathtt{mixDone}, b, \mathcal{O}')$ appeared in the leakage port of $\mathcal{F}^{\mathtt{OMix}}$. This means that $\mathsf{Audit}^*_{\mathtt{OMix}}$ returns true either in step 3a or in step 4.

Now, assume the conditions of the step 3 holds. Then, in the view of the environment there is the list of encoded messages $A \leftarrow \mathsf{Sort}(\ \langle X_i \rangle_{i \in \mathbf{C}^S}, \langle \mathsf{Encode}(\mathsf{pub}, \mathtt{M}_i^O) \rangle_{i \in [n] \setminus \mathbf{C}^S}\ )$, which are exactly the outputs of the ideal functionality $\mathcal{F}^{\mathtt{Mix}}$, and by following the steps of the protocol (which S simulates) the simulated ideal functionality `sMix` sends the messages $(\mathtt{unveil}, \mathsf{sec})$. Hence, because of the correctness of the decoding procedure Decode^X, the algorithm Audit does not outputs false as consequence of the check in steps 3b and 3c.

At this point, we are only left with the possibility that Audit returns false in step 4. However, we have already shown above that $\mathsf{Audit}^*_{\mathtt{OMix}}$ returns true. Moreover, if we reached step 4, the message $(\mathtt{mixDone}, \mathtt{invalid}, \mathcal{O}')$ was sent to all the mixers (as step 2 verified), and because the protocol terminated, the simulated ideal functionality $\mathcal{F}^{\mathtt{sMix}}[\mathcal{ES}]$ sends the message $(\mathtt{mixDone}, \mathcal{O})$ with $\mathcal{O} = \mathcal{O}^*$ (the latter equality holds by construction of Π'' and due to the fact that $\mathsf{Audit}^*_{\mathtt{Mix}}$ returns true). But then it must be that both $\mathsf{Audit}^*_{\mathtt{sMix}}$ and $\mathsf{Audit}^*_{\mathtt{DKG}}$ return true in step 4, which is a contradiction as started from the assumption that Audit returns false.

The other case – Audit returns true but $\mathsf{Audit}^*_{\mathtt{Mix}}$ returns false – simply cannot happen by definition of S (as the simulator does use the ideal functionality to produce its output).

6.3 Concrete Instantiation for the Structured Mix

In this section we show that the UC-secure (sender) verifiable mix-net[10] of Wikström [51] can be framed as a structured verifiable mixing.

Wikström shows a mixing protocol with resources the bulletin board BB, an ideal El Gamal distributed key generation[11] functionality $\mathcal{F}^{\mathsf{KG}}$ and an ideal zero-knowledge functionality $\mathcal{F}^{\mathsf{ZK}}$ for proving knowledge of the plaintexts (for the senders) and for the knowledge of a special kind of shuffle (for the mixers). Let us call $\Pi^{\mathtt{WikO5}}$ such protocol.

Briefly, the mixers compute and share the secret and public parameter for a special form of El Gamal encryption scheme using the functionality $\mathcal{F}^{\mathsf{KG}}$. The senders, given the public key pk, encrypt their messages and send them to the bulletin board and later they create a zero-knowledge proof of knowledge of the plaintext using the ideal functionality $\mathcal{F}^{\mathsf{ZK}}$. At this point, the mixers decrypt and sort the lists of ciphertexts and provide zero-knowledge proofs of knowledge of correct decryption and sorting (again using the ideal functionality).

We recall that for our compiler we need to realize a weaker mixing functionality where there are no senders and where the inputs are given encrypted to the functionality. The former property implies that we can cut from the protocol the input submission phase executed by the senders, the latter property implies that we can avoid the phase where the senders prove in

[10] The definition of Wikström for the ideal functionality of mixing is slightly different, but it is not hard to see that it is compatible with ours. Specifically, the ideal functionality defined there is strong enough to realizes our ideal $\mathcal{F}^{\mathtt{Mix}}$ functionality.

[11] The functionality accepts as input tuple of public and secret keys, and distributes the public keys and unveil the secret keys of the parties in case of agreement of the majority.

(UC) zero-knowledge to the mixers the knowledge of the plaintexts. In fact, the simulator does not have to extract the inputs from the malicious senders, since the inputs of the functionality are the ciphertexts themselves which are given "on the clear" by the senders.

The protocol of Wikström uses the resource $\mathcal{F}^{\mathsf{KG}}$ which has a special command to reveal the secret key of a mixer in case of misbehavior. We can use this special command to easily implement the command **Unveil** needed for sMix.

Notice that the encoding scheme $\mathcal{ES}$ of a structured Mix needs to be CCA2 secure, while the underlying encoding scheme in the protocol of Wikström is El Gamal which is only CPA-secure. The easiest solution to this problem is to consider a construction *à la* Naor-Young [39,44], where the encoded messages (X, Y) is such that X is an El Gamal ciphertext and Y is a SE-NIZK of knowledge of the plaintext.

Finally, we notice that the protocol $\Pi^{\mathtt{Wik05}}$ could be made auditable (according to our model) if the underlying resource $\mathcal{F}^{\mathsf{ZK}}$ is auditable. We summarize the discussion above in the following informal theorem:

Theorem 8 (Informal). *There exists a* $\mathcal{SVM} = (\mathcal{ES}, \Pi, \mathsf{Audit})$ *that is a* $\mathcal{F}^{\mathsf{KG}}, \mathcal{F}^{\mathsf{ZK}}, \mathcal{F}^{\mathsf{CRS}}$*-secure structured VMix protocol where the* X*-part of the encoded message* (X, Y) *is made only of 2 groups elements.*

As a final note, our compiler can also work with verifiable mixing schemes that are not UC-secure, provided that the protocols are proven secure in the real/ideal security paradigm of Goldreich, Micali and Wigderson [27]. In fact, as noted by Canetti (c.f. [11], pag. 70), we can consider the ideal functionality $\mathcal{F}_{\mathsf{NC}}$ that *wraps* a protocol into a non-concurrent and isolated environment. The end result, applying theorem 7, is a scheme with resource $\mathcal{F}_{\mathsf{NC}}$ where, informally speaking, the verifiable mixing, in case of invalid branch, must be computed on an "*isolated network*".

7 Efficiency Evaluations

We provide an estimation of costs for the instantiation of our compiler with our optimistic protocol of Section 5 and the UC-secure mix-net in [51] as structured mixing. Costs are parametrized by the number of senders, n, and the number of mixers, m. The encoding scheme in [51] is ElGamal with a ZK proof of knowledge of plaintext; this means that we can instantiate our RCCA scheme with $\ell = 2$. Our evaluations focus on the costs of mixing and verification in the optimistic case, and consider an instantiation over an elliptic-curve prime order group.

Every mixer must compute n authenticated re-randomizations and one proof for $\mathcal{NIZK}_{\mathtt{mx}}$. This requires a total of $8n$ exponentiations that can be computed offline, and $18n + 8$ exponentiations and $6n + 3$ (much cheaper) multiplications to be computed online. Moreover, the mixer posts $\lambda(22n + 11)$ bits on the BB. Precisely, for every ciphertext (consisting of 11 group elements), re-randomization requires 8 exponentiations to create the random masks (which can be computed offline), 10 full exponentiations to compute $s \cdot \bar{Y}$ and $t \cdot \bar{Y}$, and 8 exponentiations with a shorter exponent $\kappa \in \mathbb{Z}_{2^\lambda}$, which essentially count half in an elliptic-curve implementation. Creating the proof (consisting of 3 group elements and 2 elements in $\mathbb{Z}_q$) requires 5 exponentiations, plus $6n + 3$ group multiplications and 3 exponentiations for the checksum.

A verifier must read $m\lambda(22n + 11) + 16n\lambda$ bits from the BB, check all the senders NIZKs, compute m checksums for the lists of ciphertexts and check the mixers proofs. This requires $10n + 8m$ exponentiations and about $3nm$ group multiplications. Precisely, verifying one sender NIZK requires 10 exponentiations; computing m checksums costs $3m$ exponentiations and $2(nm + n + m)$ multiplications, and verifying the mixer NIZKs costs $5m$ exponentiations.

A closer look at the protocol shows that several of the exponentiations above are for fixed bases in the public parameters, and thus can be sped up (approximately by a factor 4.5) using

precomputation techniques (c.f., [1]). If we use a full exponentiation on an ECC group as time unit, these optimizations show that a mixer in our protocol needs a total of $11n$ exponentiations ($1.8n$ offline, 9.2 online), whereas a verifier needs about $5.3n + 4.1m$ exponentiations.

For a comparison, we consider the most efficient UC-secure mix-net proposed by Wikström in [51]. We observe that a precise comparison is difficult as the estimations in [51] are given for specific groups (a prime order group and an RSA group) and are presented using exponentiations in prime order groups as time unit. We computed estimations for the [51] protocol, considering optimizations such as multi-exponentiations and precomputations, and an instantiation of their prime order groups with ECC. Using an ECC exponentiation as time unit, we have that a mixer and verifier in [51] must do the equivalent of at least $30n$ and $22nm$ exponentiations.[12] In terms of rounds of communications our protocol needs one extra (parallel) broadcast round in which the mixers reveal the authentication keys, plus the rounds needed for the distributed decryption of the ciphertexts returned by the last mixer. However, in the protocol of Wikstrom every mixer must post in the BB at least $6n\lambda + 6n|N|$ bits (where $|N|$ is the bit size of an RSA modulus for security parameter λ), which for example are $19200n$ bits for $\lambda = 128$. In contrast, in our protocol each mixer posts $2816n + 1408$ bits, that is about 7 times shorter.

8 Conclusions and Open Problems

We revisited the notion of optimistic mixing showing new feasibility results for this, previously abandoned, research direction. Our contributions give evidences that the optimistic approach can lead to new practical schemes. Below, we identify a number of interesting problems that are left open by our work. We believe that our contributions, by reviving the interest of the research community, are an interesting starting point.

First, in our protocol an attacker might decide to invalidate the execution of an optimistic protocol only to slow down the mixing process (let us call this a "trolling attack"). We could consider a more granular definition of optimistic security that protects against these attacks for a specific subset of the parties. For example, we could look for protocols where trolling attacks are ruled out for senders (e.g., by asking them to send a NIZK proof of correct decryption). Moreover, in the same vein of covert secure MPC [7], we could look for protocols with an "investigation phase" where the dishonest party is identified.

Although our construction uses specific properties of our re-randomizable RCCA encryption scheme, our paradigm is applicable to other RCCA PKE schemes. Ideally, a more abstract construction would reduce the problem of constructing efficient OMix protocols to the (conceptually easier) problem of constructing re-randomizable RCCA scheme.

Our protocol makes use of a *local* random oracle functionality. Recently, Camenisch *et al.*[10] proposed a definition of global random oracle functionality which allows for very practical implementation of many cryptographic primitives such as CCA2 PKE, signatures and commitment schemes. It would be interesting to extend their analysis to rerandomizable RCCA PKE in the random oracle model. In particular, provided that the RCCA PKE is secure with a global random oracle, a minimal modification to our OMix protocol would result in a secure OMix in the Generalized UC [13] with a *programmable and observable* random oracle (see [10] for more details) as global setup assumption.

Finally, the main bottleneck of our construction is the distributed decryption of the RCCA ciphertexts (recall that our OMix protocol uses as ideal resource the functionality $\mathcal{F}^{\mathsf{Dec}}$). Constructing new re-randomizable RCCA PKEs with efficient distributed decryption, or giving a practical protocol for distributed decryption of our scheme are indeed the most important open problems.

[12] Our estimations are quite generous and for example do not include the time to generate n 100-bit primes.

9 Acknowledgements

Research leading to these results has been supported by the Spanish Ministry of Economy under the projects Dedetis (ref. TIN2015-70713-R) and Datamantium (ref. RTC-2016-4930-7), and by the Madrid Regional Government under project N-Greens (ref. S2013/ICE-2731).

We would like to thank Victor Mateu who suggested us to look at the original paper on Optimistic Mixing of Golle *et al.*.

References

1. Miracl cryptographic library user guide. https://libraries.docs.miracl.com/miracl-explained/benchmarks.
2. M. Abe. Universally verifiable mix-net with verification work independent of the number of mix-servers. In K. Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 437–447. Springer, Heidelberg, May / June 1998.
3. M. Abe. Mix-networks on permutation networks. In K.-Y. Lam, E. Okamoto, and C. Xing, editors, *ASIACRYPT'99*, volume 1716 of *LNCS*, pages 258–273. Springer, Heidelberg, Nov. 1999.
4. M. Abe and F. Hoshino. Remarks on mix-network based on permutation networks. In K. Kim, editor, *PKC 2001*, volume 1992 of *LNCS*, pages 317–324. Springer, Heidelberg, Feb. 2001.
5. M. Abe and H. Imai. Flaws in some robust optimistic mix-nets. In R. Safavi-Naini and J. Seberry, editors, *ACISP 03*, volume 2727 of *LNCS*, pages 39–50. Springer, Heidelberg, July 2003.
6. B. Adida and D. Wikström. Offline/online mixing. In L. Arge, C. Cachin, T. Jurdzinski, and A. Tarlecki, editors, *ICALP 2007*, volume 4596 of *LNCS*, pages 484–495. Springer, Heidelberg, July 2007.
7. Y. Aumann and Y. Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. In S. P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 137–156. Springer, Heidelberg, Feb. 2007.
8. C. Baum, I. Damgård, and C. Orlandi. Publicly auditable secure multi-party computation. In M. Abdalla and R. D. Prisco, editors, *SCN 14*, volume 8642 of *LNCS*, pages 175–196. Springer, Heidelberg, Sept. 2014.
9. S. Bayer and J. Groth. Efficient zero-knowledge argument for correctness of a shuffle. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 263–280. Springer, Heidelberg, Apr. 2012.
10. J. Camenisch, M. Drijvers, T. Gagliardoni, A. Lehmann, and G. Neven. The wonderful world of global random oracles. In J. B. Nielsen and V. Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 280–312. Springer, Heidelberg, Apr. / May 2018.
11. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. http://eprint.iacr.org/2000/067.
12. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, Oct. 2001.
13. R. Canetti, Y. Dodis, R. Pass, and S. Walfish. Universally composable security with global setup. In S. P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 61–85. Springer, Heidelberg, Feb. 2007.
14. R. Canetti, H. Krawczyk, and J. B. Nielsen. Relaxing chosen-ciphertext security. In D. Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 565–582. Springer, Heidelberg, Aug. 2003.
15. M. Chase, M. Kohlweiss, A. Lysyanskaya, and S. Meiklejohn. Malleable proof systems and applications. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 281–300. Springer, Heidelberg, Apr. 2012.
16. D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, Feb. 1981.
17. R. Cramer, I. Damgård, and J. B. Nielsen. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015.
18. R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In L. R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer, Heidelberg, Apr. / May 2002.
19. I. Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In B. Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 418–430. Springer, Heidelberg, May 2000.
20. A. Escala, G. Herold, E. Kiltz, C. Ràfols, and J. Villar. An algebraic framework for Diffie-Hellman assumptions. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Heidelberg, Aug. 2013.
21. S. Faust, M. Kohlweiss, G. A. Marson, and D. Venturi. On the non-malleability of the Fiat-Shamir transform. In S. D. Galbraith and M. Nandi, editors, *INDOCRYPT 2012*, volume 7668 of *LNCS*, pages 60–79. Springer, Heidelberg, Dec. 2012.

22. U. Feige, D. Lapidot, and A. Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *31st FOCS*, pages 308–317. IEEE Computer Society Press, Oct. 1990.
23. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, Aug. 1987.
24. J. Furukawa and K. Sako. An efficient scheme for proving a shuffle. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 368–387. Springer, Heidelberg, Aug. 2001.
25. R. Gennaro, M. O. Rabin, and T. Rabin. Simplified VSS and fact-track multiparty computations with applications to threshold cryptography. In B. A. Coan and Y. Afek, editors, *17th ACM PODC*, pages 101–111. ACM, June / July 1998.
26. O. Goldreich. A uniform-complexity treatment of encryption and zero-knowledge. *Journal of Cryptology*, 6(1):21–53, 1993.
27. O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In *27th FOCS*, pages 174–187. IEEE Computer Society Press, Oct. 1986.
28. S. Goldwasser and Y. Lindell. Secure multi-party computation without agreement. *Journal of Cryptology*, 18(3):247–287, July 2005.
29. P. Golle, M. Jakobsson, A. Juels, and P. F. Syverson. Universal re-encryption for mixnets. In T. Okamoto, editor, *CT-RSA 2004*, volume 2964 of *LNCS*, pages 163–178. Springer, Heidelberg, Feb. 2004.
30. P. Golle, S. Zhong, D. Boneh, M. Jakobsson, and A. Juels. Optimistic mixing for exit-polls. In Y. Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 451–465. Springer, Heidelberg, Dec. 2002.
31. J. Groth. A verifiable secret shuffle of homomorphic encryptions. In Y. Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 145–160. Springer, Heidelberg, Jan. 2003.
32. J. Groth. Rerandomizable and replayable adaptive chosen ciphertext attack secure cryptosystems. In M. Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 152–170. Springer, Heidelberg, Feb. 2004.
33. J. Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In X. Lai and K. Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 444–459. Springer, Heidelberg, Dec. 2006.
34. J. Groth. A verifiable secret shuffle of homomorphic encryptions. *Journal of Cryptology*, 23(4):546–579, Oct. 2010.
35. J. Groth and Y. Ishai. Sub-linear zero-knowledge argument for correctness of a shuffle. In N. P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 379–396. Springer, Heidelberg, Apr. 2008.
36. M. Jakobsson and D. M'Raïhi. Mix-based electronic payments. In S. E. Tavares and H. Meijer, editors, *SAC 1998*, volume 1556 of *LNCS*, pages 157–173. Springer, Heidelberg, Aug. 1999.
37. B. Libert, T. Peters, and C. Qian. Structure-preserving chosen-ciphertext security with shorter verifiable ciphertexts. In S. Fehr, editor, *PKC 2017, Part I*, volume 10174 of *LNCS*, pages 247–276. Springer, Heidelberg, Mar. 2017.
38. Y. Lindell, A. Lysyanskaya, and T. Rabin. On the composition of authenticated byzantine agreement. In *34th ACM STOC*, pages 514–523. ACM Press, May 2002.
39. M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*, pages 427–437. ACM Press, May 1990.
40. C. A. Neff. A verifiable secret shuffle and its application to e-voting. In *ACM CCS 01*, pages 116–125. ACM Press, Nov. 2001.
41. C. Park, K. Itoh, and K. Kurosawa. Efficient anonymous channel and all/nothing election scheme. In T. Helleseth, editor, *EUROCRYPT'93*, volume 765 of *LNCS*, pages 248–259. Springer, Heidelberg, May 1994.
42. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In J. Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, Heidelberg, Aug. 1992.
43. M. Prabhakaran and M. Rosulek. Rerandomizable RCCA encryption. In A. Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 517–534. Springer, Heidelberg, Aug. 2007.
44. A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th FOCS*, pages 543–553. IEEE Computer Society Press, Oct. 1999.
45. K. Sako and J. Kilian. Receipt-free mix-type voting scheme - a practical solution to the implementation of a voting booth. In L. C. Guillou and J.-J. Quisquater, editors, *EUROCRYPT'95*, volume 921 of *LNCS*, pages 393–403. Springer, Heidelberg, May 1995.
46. C.-P. Schnorr. Efficient identification and signatures for smart cards. In G. Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer, Heidelberg, Aug. 1990.
47. V. Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. http://eprint.iacr.org/2004/332.
48. B. Terelius and D. Wikström. Proofs of restricted shuffles. In D. J. Bernstein and T. Lange, editors, *AFRICACRYPT 10*, volume 6055 of *LNCS*, pages 100–113. Springer, Heidelberg, May 2010.

49. D. Wikström. Five practical attacks for "optimistic mixing for exit-polls". In M. Matsui and R. J. Zuccherato, editors, *SAC 2003*, volume 3006 of *LNCS*, pages 160–175. Springer, Heidelberg, Aug. 2004.
50. D. Wikström. A universally composable mix-net. In M. Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 317–335. Springer, Heidelberg, Feb. 2004.
51. D. Wikström. A sender verifiable mix-net and a new proof of a shuffle. In B. K. Roy, editor, *ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 273–292. Springer, Heidelberg, Dec. 2005.
52. D. Wikström. Simplified submission of inputs to protocols. In R. Ostrovsky, R. D. Prisco, and I. Visconti, editors, *SCN 08*, volume 5229 of *LNCS*, pages 293–308. Springer, Heidelberg, Sept. 2008.
53. D. Wikström. A commitment-consistent proof of a shuffle. In C. Boyd and J. M. G. Nieto, editors, *ACISP 09*, volume 5594 of *LNCS*, pages 407–421. Springer, Heidelberg, July 2009.
54. D. Wikström. Verificatum, 2010. https://www.verificatum.com.

A NIZK Proof Systems

In this section we show how to instantiate the NIZK proof systems used in the protocol of Sec. 5.

A.1 Instatiations in the CRS+NPRO Model

The Mixers NIZK $\mathcal{NIZK}_{\mathtt{mx}}$. Consider the NP relation:

$$\mathcal{R}_l = \left\{((\bar{S}, \bar{R}), w) \in \mathbb{G}^{2l} \times \mathbb{Z}_q : \ \bar{R} = w \cdot \bar{S}\right\}$$

where $(\bar{S}, \bar{R})$ is the instance and w the witness. We show a NIZK $\mathcal{NIZK}_l$ for $\mathcal{R}_l$. Notice that $\mathcal{R}_{\mathtt{mx}} = \mathcal{R}_3$, and thus $\mathcal{NIZK}_{\mathtt{mx}}$ is $\mathcal{NIZK}_l$ with $l = 3$.

$\mathsf{Init}_{\mathtt{mx}}(1^\lambda, \mathtt{prm})$**:** sample $\mathbf{v} \leftarrow_\$ \mathbb{Z}_q^2$ and two hash functions H, H', output $\mathsf{crs} = (H, H', [\mathbf{v}])$.[13]
$\mathsf{P}_{\mathtt{mx}}(\mathsf{crs}, (L, \bar{S}, \bar{R}), w)$**:** sample $a, r \leftarrow_\$ \mathbb{Z}_q$ and compute: $\bar{A} \leftarrow a \cdot \bar{S} \in \mathbb{G}^l$, $A' = (H'(\bar{A}), r)^\top \cdot [\mathbf{v}] \in \mathbb{G}$, $\beta \leftarrow H(L, \bar{S}, \bar{R}, A') \in \mathbb{Z}_q$, $\gamma \leftarrow a + \beta \cdot w$. Return $\Pi = (\bar{A}, r, \gamma)$.
$\mathsf{V}_{\mathtt{mx}}(\mathsf{crs}, (L, \bar{S}, \bar{R}), \Pi)$**:** output 1 iff $\gamma \cdot \bar{S} = \bar{A} + H(L, \bar{S}, \bar{R}, A') \cdot \bar{R}$, with $A' = (H'(\bar{A}), r)^\top \cdot [\mathbf{v}]$.

Efficiency. A proof consists of l elements of $\mathbb{G}$ plus 2 values in $\mathbb{Z}_q$. The prover must perform $l + 2$ exponentiations, while the verifier performs $2l + 2$ exponentiations.[14]

Theorem 9. *If the discrete logarithm assumption holds in $\mathbb{G}$, H' is collision-resistant, and H is modeled as a non-programmable random-oracle, then $\mathcal{NIZK}_l$ is a NIZK.*

The theorem follows by combining a series of results given in the following section. In particular, the NIZK above is obtained by applying a transformation by Damgård [19] (and then the Fiat-Shamir [23] transform) to a simple sigma protocol for relation $\mathcal{R}_l$.

The Senders NIZK $\mathcal{NIZK}_{\mathtt{sd}}$. We give a simulation f-extractable NIZK for relation $\mathcal{R}_{\mathtt{sd}}$. $\mathcal{R}_{\mathtt{sd}} = \left\{(([\mathbf{g}, a], \bar{W}), (w, R)) \in \mathbb{G}^6 \times \mathbb{Z}_q^2 : \ \bar{W} = w \cdot [\mathbf{g}, a] + (0, 0, R)\right\}$. In our case f-extractability says that we can extract only the witness component R, which is sufficient for our purposes.

$\mathsf{Init}_{\mathtt{sd}}(1^\lambda, \mathtt{prm})$**:** sample $\mathbf{u}, v \leftarrow_\$ \mathbb{Z}_q^2$ and two hash functions H, H', output $\mathsf{crs} = (H, H', [\mathbf{u}, \mathbf{v}])$.
$\mathsf{P}_{\mathtt{sd}}(\mathsf{crs}, (L, [\mathbf{g}, a], \bar{W}), (w, R))$**:** sample $a, r \leftarrow_\$ \mathbb{Z}_q$ and compute: $\bar{E} \leftarrow w \cdot [\mathbf{u}] + (R, 0)$, $\bar{S} \leftarrow ([\mathbf{g}], [a] - [u_1], -[u_2])$, $\bar{R} \leftarrow (\bar{W}, 0) - (0, 0, \bar{E})$. $\bar{A} \leftarrow a \cdot \bar{S} \in \mathbb{G}^4$, $A' = (H'(\bar{A}), r)^\top \cdot [\mathbf{v}] \in \mathbb{G}$, $\beta \leftarrow H(L, \bar{S}, \bar{R}, A') \in \mathbb{Z}_q$, $\gamma \leftarrow a + \beta \cdot w$. Return $\Pi = (\bar{A}, r, \gamma, \bar{E})$.
$\mathsf{V}_{\mathtt{sd}}(\mathsf{crs}, (L, [\mathbf{g}, a], \bar{W}), \Pi)$**:** compute $\bar{S} \leftarrow ([\mathbf{g}], [a] - [u_1], -[u_2])$, $\bar{R} \leftarrow (\bar{W}, 0) - (0, 0, \bar{E})$, and output 1 iff $\gamma \cdot \bar{S} = \bar{A} + H(L, \bar{S}, \bar{R}, A') \cdot \bar{R}$, with $A' = (H'(\bar{A}), r)^\top \cdot [\mathbf{v}]$.

[13] We note that $[v]$ is simply the key of a Pedersen commitment.
[14] We ignore the costs of computing hash functions, group multiplications and $\mathbb{Z}_q$ operations as they are negligible in comparison.

Efficiency. A proof consists of 6 elements of $\mathbb{G}$ plus 2 values in $\mathbb{Z}_q$. The prover (resp. verifier) must perform 8 (resp. 10) exponentiations.

Theorem 10. *If the DDH assumption holds in $\mathbb{G}$, H' is collision-resistant, and H is modeled as a non-programmable random-oracle, then $\mathcal{NIZK}_{\mathtt{sd}}$ is a f-simulation-extractable NIZK with distributional zero-knowledge (i.e., it achieves zero-knowledge when the elements $[\mathbf{g}, a], w, R$ of the relation come from a uniform distribution).*

The theorem follows by combining a series of results given in the following section. In a nutshell, this NIZK is a sigma protocol in which one encrypts the witness component R using ElGamal, and then use $\mathcal{NIZK}_4$ (with simulation-soundness) to show that the same R is encrypted with ElGamal and in $\bar{W}$. The sigma protocol we use has a straight-line simulator (Damgaard [19]). Moreover we use an optimization that consists in using w as randomness for ElGamal (relying on the fact that this witness comes from a uniform distribution). Finally, Fiat-Shamir [23] is applied to make it non-interactive.

A.2 How to obtain $\mathcal{NIZK}_{\mathtt{mx}}$ and $\mathcal{NIZK}_{\mathtt{sd}}$

We first recall the notion of Σ-protocols.

Definition 15 (Sigma protocol). *A protocol Π between a prover $\mathcal{P}_P$ and a verifier $\mathcal{P}_V$ is said to be a Σ-protocol for a relation $\mathcal{R}$ if:*

1. *Π is a three move protocol where the prover starts and the second message is uniformly random over the challenge space. (We will parse a transcript of an execution as (α, β, γ).) The protocol is complete, meaning that if the prover and verifier follow the protocol on input $(x, w) \in \mathcal{R}$ then the verifier always accepts.*
2. *For any x and any pair of accepting transcripts α, β, γ and α, β', γ' where $\beta \neq \beta'$ one can efficiently compute w such that $(x, w) \in \mathcal{R}$. This is often called* special soundness *property.*
3. *There exists a simulator S that upon input β and an instance x outputs a transcript α, β, γ distributed as a real transcript conditioned on β. This is often called* special honest-verifier zero-knowledge *(HV-ZK) property.*

In this section we give a SS-NIZK proof system for any relation $\mathcal{R}$ in the CRS+NPRO model based on the existence of a Σ-protocol for $\mathcal{R}$. Specifically, to get the non-programability property, we consider a non-interactive version (using the Fiat-Shamir methodology [23]) of the concurrent ZK protocol based on Σ-protocols of Damgård [19].

We first notice that Fiat-Shamir-based NIZK with labels are simulation-sound. In [21], Faust *et al.* showed that Fiat-Shamir NIZK based on Σ-protocols can be proved simulation sound in the programmable random oracle model based on a special property of sigma protocol called *quasi unique responses*. However, when considering the definition with labels, as we show below, the above property is not necessary.

The Construction of [19]. The construction is based on the notion of trapdoor-hiding commitment schemes. We recall here the syntax of the primitive and state only informally its security properties.

Definition 16 (Trapdoor-hiding commitment). *We say that $\mathcal{COM} = (\mathsf{Setup}, \mathsf{Com}, \overline{\mathsf{Com}}, \mathsf{Equiv})$ is a* trapdoor-hiding *commitment scheme if the following holds.*

1. *The algorithm Setup upon input 1^λ (and optionally parameter $\mathtt{prm}$) outputs a verification key vk and a trapdoor key td^{com};*

2. *The algorithm* Com *upon input* vk*, a message* m *and randomness* r*, outputs a commitment* Com*;*
3. *The algorithm* $\overline{\mathsf{Com}}$ *upon input* td^{com} *outputs a commitment* Com *and an equivocation value* r'*;*
4. *The algorithm* Equiv *upon input* td^{com}*, a message* m *and* r'*, outputs randomness* r *such that for any* $(Com, r') \leftarrow \overline{\mathsf{Com}}(\mathsf{td}^{com})$ *and* $r \leftarrow (\mathsf{td}^{com}, m, r')$ *then* $Com = \mathsf{Com}(vk, m, r)$*.*

We require that the commitment is **binding**, *meaning that given only* vk *is hard to find two tuples* (m_0, r_0) *and* (m_1, r_1) *such that* $\mathsf{Com}(vk, m_0; r_0) = \mathsf{Com}(vk, m_1; r_1)$*, and* **trapdoor-hiding**, *meaning that for all* $(vk, \mathsf{td}^{com}) \leftarrow \mathsf{Setup}(1^\lambda)$ *and for all* m *the distribution* (Com, r) *generated by* $Com, r' \leftarrow\!\!\$\ \overline{\mathsf{Com}}(\mathsf{td}^{com})$ *and* $r \leftarrow \mathsf{Equiv}(\mathsf{td}^{com}, m, r')$ *is indistinguishable from the distribution of* (Com, r) *generated by* Com*.*

We describe the construction of [19], let Π be a sigma protocol between a prover $\mathcal{P}_P$ and a verifier $\mathcal{P}_V$ for a relation $\mathcal{R}$, and let $\mathcal{COM} = (\mathsf{Setup}, \mathsf{Com}, \overline{\mathsf{Com}}, \mathsf{Equiv})$ be a trapdoor-hiding commitment scheme. Consider the following protocol Π' in the CRS model

- Let Init be the CRS generation algorithm that upon input 1^λ, sample $vk, \mathsf{td}^{com} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$ and set the common reference string to be vk; (If the protocol Π is in the CRS model, then also the common reference string of Π is sampled.)
- Let $\mathcal{P}'_P$ be the prover of a new Σ-protocol that:
  1. upon input x, w and vk and randomness $r, r' \leftarrow\!\!\$\ \{0,1\}^\lambda$, computes $\alpha \leftarrow \mathcal{P}_P(x, w; r)$, $\alpha' \leftarrow \mathsf{Com}(vk, \alpha; r')$, and outputs as first message the value α';
  2. upon input x, w, randomness r, r', and challenge β', computes $\gamma \leftarrow \mathcal{P}_P(x, w, \beta'; r)$ and output $\gamma' = (\alpha, r', \gamma)$;
- Let $\mathcal{P}'_V$ be the verifier that upon a transcript α', β', γ', parse $\gamma' = \alpha, r', \gamma$, and outputs 1 if and only if (1) $\alpha' = \mathsf{Com}(vk, \alpha; r')$ and (2) $\mathcal{P}_V(\alpha, \beta', \gamma) = 1$.

Theorem 11 ([19]). *The protocol* Π' *described above is a concurrent zero-knowledge proof system for* $\mathcal{R}$ *in the common reference string model. In particular, there exists a algorithm* $\overline{\mathsf{Init}}'$ *and a* straight-line *(namely, that does not use rewinding technique) simulator* S' *that, with oracle access to an adversary* A*, produces a view that it is indistinguishable from an interaction of the adversary with the honest prover.*

Remark 3. The theorem above holds also when the adversary can chose the instance x, w as a function of the verification key vk. In fact, the trapdoor-hiding property holds for all $(vk, \mathsf{td}^{com}) \leftarrow \mathsf{Setup}(1^\lambda)$ and for all m.

The Fiat-Shamir transform. From a Sigma protocol for relation $\mathcal{R}$ as the one obtained from the previous theorem, we can build a NIZK scheme with labels $\mathcal{NIZK} = (\mathsf{Init}, \mathsf{P}, \mathsf{V})$ for $\mathcal{R}$ where: (1) Init is exactly as described in the protocol above, (2) the prover P upon input x, w and a tag L:

1. Samples randomness r and produces $\alpha' \leftarrow \mathcal{P}'_P(vk, x, w; r, r')$,
2. Computes $\beta' \leftarrow \mathsf{RO}(x \| L \| \alpha)$
3. Computes $\gamma' \leftarrow \mathcal{P}'_P(x, w, \beta'; r)$ and return the proof (α, γ);

(3) the verifier V re-computes β' given x, L, α using the random oracle, and outputs what $\mathcal{P}'_V(\alpha', \beta', \gamma')$ does.

Theorem 12. *The scheme* $\mathcal{NIZK}$ *described above is a NIZK with adaptive multi-theorem zero-knowledge in the Non-Programmable Random Oracle model.*

Proof (Proof Sketch.). The proof is rather easy. The algorithm $\overline{\mathsf{Init}}$ for $\mathcal{NIZK}$ is exactly the same provided by theorem 11. We describe a simulator $\mathsf{S}^{\mathsf{NPRO}}$. Let S' be the simulator given by the theorem 11. We define our new simulator to be the machine that upon input an instance x and a tag L, runs the simulator S' on input x, and upon the first message α from S', the simulator $\mathsf{S}^{\mathsf{NPRO}}$ computes $\mathtt{RO}(x\|L\|\alpha)$ as prescribed by the definition of the protocol. We notice that this new simulator does not need to program the random oracle. The theorem holds by a simple hybrid argument over the number of invocations of $\mathcal{SIM}$, reducing to the zero-knowledge of the interactive construction.

We recall the definition of simulation soundness, this property is strictly stronger than adaptive soundness (Def. 1) necessary to instantiate the NIZK $\mathcal{NIZK}_{\mathtt{mx}}$ in Sec. 5.

Definition 17 (simulation soundness, [44]). *Let $\mathcal{NIZK}$ be a non-interactive argument system for a relation $\mathcal{R}$. We say that $\mathcal{NIZK}$ is* simulation sound *(SS) with labels if the following holds:*

(i) There exists an algorithm $\overline{\mathsf{Init}}$ that outputs a CRS crs and a simulation trapdoor td^s.
(ii) For all PPT adversaries A, we have $\mathsf{P}[\mathsf{A}$ wins$] \in \mathtt{negl}(\lambda)$ in the following game:
- *The challenger runs $(\mathsf{crs}, \mathsf{td}^s, \mathsf{td}^e) \leftarrow \overline{\mathsf{Init}}(1^\lambda)$, and gives crs to A.*
- *A is given access to the simulation oracle $\mathcal{SIM}^*$, which it can query adaptively.*
- *A outputs a tuple $((x^*, L^*), \Pi^*)$. We say that A wins if: (a) the pair (x^*, L^*) was not queried to the simulation oracle; (b) $\mathsf{V}(\mathsf{crs}, (x^*, L^*), \Pi^*) = 1$; and (c) $\forall r : (x^*, r) \notin \mathcal{R}$. The simulation oracle $\mathcal{SIM}(\mathsf{td}^s, \cdot)$ responds to queries of the form (x, L) by outputting a simulated argument $\mathsf{Sim}(\mathsf{td}^s, (x, L))$.*

Theorem 13 (simulation-soundness of Fiat-Shamir in the RO). *The scheme $\mathcal{NIZK}$ described above is simulation-sound in the (programmable) RO model.*

Proof (Proof Sketch.). Let A be an adversary that interacts with the oracle $\mathcal{SIM}^*$ for $q \in \mathbb{N}$ times and with the random oracle $\mathtt{RO}$. The adversary forwards a queries (x_i, L_i) for $i \in [q]$ to $\mathcal{SIM}^*$ which replies with (α_i^*, γ_i^*). Successively, the adversary outputs a tuple $\tilde{x}, \tilde{L}$ and a NIZK proof $(\tilde{\alpha}, \tilde{\gamma})$. There are two cases, either $\tilde{x} = x_i$ and $\tilde{\alpha} = \alpha_i^*$ for some $i \in [q]$, in this case the labels $\tilde{L}$ and L_i are different therefore $\mathtt{RO}(x_i\|L_i\|\alpha) \neq \mathtt{RO}(x_i\|\tilde{L}\|\alpha)$, so we can extract the witness using the special soundness of the interactive protocol. On the other hand it could be that $\tilde{x}$ is not in $\{x_i\}_{i\in[q]}$, in this case we can, via standard rewinding technique and by programming the random oracle again reduce to special soundness of the interactive protocol.

We notice that the proof of the simulation-soundness does program the random oracle model, however, we use programmability only to prove soundness but we do not need to program explicitly the random oracle when utilizing the NIZK $\mathcal{NIZK}$ as a primitive in a bigger protocol.

A.3 A Sigma protocol for $\mathcal{R}_{\mathtt{mx}}$

Consider the relationship $\mathcal{R}_l = \left\{((\bar{S}, \bar{R}), w) \in \mathbb{G}^{2l} \times \mathbb{Z}_q : \ \bar{R} = w \cdot \bar{S}\right\}$, we notice $\mathcal{R}_{\mathtt{mx}} = \mathcal{R}_3$. The protocol below is a generalization to vectors of the classical Schnorr's protocol [46]. Consider the Σ-protocol with parties $\mathcal{P}_P^l, \mathcal{P}_V^l$ for the relation $\mathcal{R}_l$:

- The party $\mathcal{P}_P^l$ upon input $(\bar{S}, \bar{R}), w$, a tag L, the parameters of the group $\mathtt{prm}$:
  1. Sample randomness $a \leftarrow_\$ \mathbb{Z}_q$ and compute $\alpha \leftarrow a \cdot \bar{S}$
  2. Upon challenge $\beta \in \mathbb{Z}_q$ compute $\gamma \leftarrow a + \beta \cdot w$
- The party $\mathcal{P}_V^l$ upon input $(\bar{S}, \bar{R})$, a tag L, the parameters of the group $\mathtt{prm}$ and a transcript α, β, γ outputs 1 if and only if $\gamma \cdot \bar{S} = \alpha + \beta \cdot \bar{R}$.

Theorem 14. *The protocol described above is a sigma protocol.*

We can derive the NIZK proof system $\mathcal{NIZK}_{\mathtt{mx}}$ presented at the beginning of this section using the above results, where the trapdoor-hiding commitment is instantiated with the classical Pedersen's commitment scheme [42] based on discrete log. Moreover, although a Pedersen's commitment allows one to commit values in $\mathbb{Z}_q$, we can commit to $\alpha \in \mathbb{G}^l$ using a collision resistant hash function from $\mathbb{G}^l$ to $\mathbb{Z}_q$ and storing the hash key in the verification key of the commitment.

A.4 Non-interactive zero-knowledge proof system for $\mathcal{R}_{\mathtt{sd}}$

We show a simulation f-extractable NIZK for $\mathcal{R}_{\mathtt{sd}}$. Here we present a construction obtained in a modular way from $\mathcal{NIZK}_4$ for the relation $\mathcal{R}_4$, and the ElGamal encryption scheme, that is used to get extractability. More in details, $\mathcal{NIZK}_{\mathtt{sd}} = (\mathsf{Init}_{\mathtt{sd}}, \mathsf{P}_{\mathtt{sd}}, \mathsf{V}_{\mathtt{sd}})$ is defined as follow:

- $\mathsf{Init}_{\mathtt{sd}}(1^\lambda, \mathtt{prm})$: sample an ElGamal public key $\bar{H} = [h_1, h_2] \leftarrow_\$ \mathbb{G}^2$, and compute $\mathsf{crs}_4 \leftarrow_\$ \mathsf{Init}_4(1^\lambda)$ of $\mathcal{NIZK}_4$, and output the tuple $\mathsf{crs}_{\mathtt{sd}} = (\bar{H}, \mathsf{crs}_4)$;
- $\mathsf{P}_{\mathtt{sd}}(\mathsf{crs}_{\mathtt{sd}}, (L, [\mathbf{g}, a], \bar{W}), (w, R))$: upon input an instance $[\mathbf{g}, a], \bar{W}$ and a label L, a witness (w, R), and the common reference string $\mathsf{crs} = (\bar{H}, \mathsf{crs}_4)$, do the following:
  - Compute $\bar{E} \leftarrow w \cdot \bar{H} + (R, 0)$;
  - Set $\bar{S} \leftarrow [\mathbf{g}, a - h_1, -h_2]$ and set $\bar{R} \leftarrow (\bar{W}, 0) - (0, 0, \bar{E})$;
  - Let $\Pi \leftarrow_\$ \mathsf{P}_4(\mathsf{crs}, (\bar{S}, \bar{R}), w, L)$;
  - Return $\Pi' \leftarrow (\bar{E}, \Pi)$.
- $\mathsf{V}_{\mathtt{sd}}(\mathsf{crs}_{\mathtt{sd}}, (L, [\mathbf{g}, a], \bar{W}), \Pi')$: upon input the common reference string $\mathsf{crs} = (\bar{H}, \mathsf{crs}_4)$, an instance $[\mathbf{g}, a], \bar{W}$, a label L, and a proof Π', do the following:
  - Parse Π' as $(\bar{E}, \Pi)$
  - Compute $\bar{S} \leftarrow [\mathbf{g}, a - h_1, -h_2]$ and set $\bar{R} \leftarrow (\bar{W}, 0) - (0, 0, \bar{E})$;
  - Output $\mathsf{V}_4(\mathsf{crs}_4, (\bar{S}, \bar{W}), \Pi)$;

Completeness of the proof system above comes $(\bar{W}, 0) - (0, 0, \bar{E}) = w \cdot \bar{S}$.

We consider a weaker form of zero-knowledge that is still sufficient the protocol in Sec. 5. In particular, given a class of relations $\{\mathcal{R}_\lambda \subseteq \{0,1\}^\lambda \times \{0,1\}^*\}_{\lambda \in \mathbb{N}}$ and given a class of distributions $\bar{\mathcal{D}} = \{\mathcal{D}_{\lambda,n}\}_{\lambda,n \in \mathbb{N}}$ where $\mathcal{D}_{\lambda,n}$ is efficiently samplable and ranges over $(\{0,1\}^\lambda \times \{0,1\}^*)^n$ we consider a distributional definition *à la* Goldreich [26].

Definition 18 ($\bar{\mathcal{D}}$-zero-knowledge). *Let $\mathcal{NIZK} = (\mathsf{Init}, \mathsf{P}, \mathsf{V})$ be a proof system for $\{\mathcal{R}_\lambda\}_{\lambda \in \mathbb{N}}$ we say that $\mathcal{NIZK}$ is $\bar{\mathcal{D}}$-zero-knowledge if there exists a PPT algorithm $\overline{\mathsf{Init}}$ and a PPT simulator S such that for any n polynomial in λ the following distribution ensembles are indistinguishable:*

$$\mathbf{R}_{\mathcal{NIZK},\bar{\mathcal{D}}} := \left\{ \mathsf{crs}, (x_i, \mathsf{P}(\mathsf{crs}, x, w))_{i \in [n]} : \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{Init}(1^\lambda), \\ (x_0, w_0), \ldots (x_{n-1}, w_{n-1}) \leftarrow_\$ \mathcal{D}_{\lambda,n} \end{array} \right\}_{\lambda \in \mathbb{N}}$$

$$\mathbf{S}_{\mathcal{NIZK},\bar{\mathcal{D}}} := \left\{ \mathsf{crs}, (x_i, \mathsf{S}(\mathsf{td}^s, x_i))_{i \in [n]} : \begin{array}{c} \mathsf{crs}, \mathsf{td}^s \leftarrow \overline{\mathsf{Init}}(1^\lambda), \\ (x_0, w_0), \ldots (x_{n-1}, w_{n-1}) \leftarrow_\$ \mathcal{D}_{\lambda,n} \end{array} \right\}_{\lambda \in \mathbb{N}}$$

Theorem 15. *For any set $S = \{\mathtt{prm} \leftarrow \mathsf{Setup}(1^\lambda)\}_{k \in \mathbb{N}}$, let $\bar{\mathcal{U}}_S = \{\mathcal{U}_{\mathtt{prm},n}\}_{\mathtt{prm} \in S}$ where $\mathcal{U}_{\mathtt{prm},n}$ is the distribution that picks at random $[\mathbf{g}, a] \leftarrow_\$ \mathbb{G}^3$, $w_i \leftarrow_\$ \mathbb{Z}_q$ and $R_i \leftarrow_\$ \mathbb{G}$ for $i \in [n]$, and outputs the tuples $(\bar{Z}, \bar{W}_i)$ where $\bar{Z} = [\mathbf{g}, a]$ and $\bar{W}_i = w_i \cdot \bar{Z} + (0, 0, R_i)$, the proof system $\mathcal{NIZK}_{\mathtt{sd}}$ is $\bar{\mathcal{U}}_S$-zero-knowledge.*

Proof. We define the algorithm $\overline{\mathsf{Init}_{\mathtt{sd}}}$ that upon input 1^λ and the group parameters $\mathtt{prm}$ samples a public key $\bar{H} = [h_1, h_2] \leftarrow_\$ \mathbb{G}^2$ and set $\mathsf{td}^e \leftarrow h_2/h_1 \in \mathbb{Z}_q$ and computes $\mathsf{crs}, \mathsf{td}^s \leftarrow_\$ \overline{\mathsf{Init}}_4(1^\lambda)$ of $\mathcal{NIZK}_4$, and output the tuple $\mathsf{crs} = (\bar{H}, \mathsf{crs}_4)$ and the trapdoors $\mathsf{td}^s, \mathsf{td}^e$; We define the simulator. Let S_4 be the simulator for the $\mathcal{NIZK}_4$.

Simulator $\mathsf{S}(\mathsf{td}^s, (\bar{Z}, \bar{W}_i))$:

1. Sample $e_i \leftarrow_\$ \mathbb{Z}_q$ and compute $\bar{E}_i \leftarrow e_i \cdot \bar{H}$;
2. Compute $\bar{S} \leftarrow [\mathbf{g}, a - h_1, -h_2]$ and set $\bar{R}_i \leftarrow (\bar{W}_i, 0) - (0, 0, \bar{E}_i)$;
3. Compute $\mathit{\Pi}_i \leftarrow_\$ \mathsf{S}_4(\mathsf{td}^s, (\bar{S}, \bar{R}_i))$;
4. Output $(\bar{E}_i, \mathit{\Pi}_i)$.

Consider an hybrid experiment $\mathbf{H}_1$ where the elements $\bar{E}_i$ are computed as $e_i \cdot \bar{H} + (R_i, 0)$. Specifically, $\mathbf{H}_1$ is the distribution:

$$\left\{ (\mathsf{crs}, (\bar{Z}, \bar{W}), (\bar{E}, \mathit{\Pi})) : \begin{array}{l} (\bar{H}, \mathsf{crs}') \leftarrow_\$ \mathsf{Init}(1^\lambda) \\ \bar{S} \leftarrow [\mathbf{g}, a - h_1, -h_2] \\ \forall i \in [n] : w_i \leftarrow_\$ \mathbb{Z}_q, R_i \leftarrow_\$ \mathbb{G} \\ \bar{E}_i \leftarrow e_i \cdot \bar{H} + (0, R_i), \\ \bar{R}_i \leftarrow w_i \cdot ([\mathbf{g}, a, 0] - (0, 0, \bar{H})) \\ \mathit{\Pi}'_i \leftarrow \mathsf{S}_4(\mathsf{td}^s, (\bar{S}, \bar{R}_i)) \end{array} \right\}$$

Claim 23 *If DDH assumption holds over $\mathbb{G}$ chosen from the sequence S then $\mathbf{H}_1 \approx_c \mathbf{S}_{\bar{\mathcal{D}}, \mathcal{NIZK}_{\mathtt{sd}}}$.*

Proof. The proof follows in three steps. First we consider the hybrid $\mathbf{H}'$ where the values $\bar{E}_i$ are sampled uniformly in $\mathbb{G}^2$. This hybrid is computationally indistinguishable from $\mathbf{H}_1$ by the DDH assumption. Then we consider the hybrid $\mathbf{H}''$ where the values $\bar{E}_i = \bar{E}'_i + (0, R_i)$ where again $\bar{E}'_i \leftarrow_\$ \mathbb{G}^2$. The distribution $\mathbf{H}'$ and $\mathbf{H}''$ are equivalent. Finally, using again the DDH assumption we can prove that $\mathbf{H}''$ is computationally indistinguishable from $\mathbf{H}_1$.

Consider an hybrid experiment $\mathbf{H}_2$ where the elements $\bar{E}_i$ are computed as $w_i \cdot \bar{H} + (R_i, 0)$. Specifically, $\mathbf{H}$ is the distribution:

$$\left\{ (\mathsf{crs}, (\bar{Z}, \bar{W}), (\bar{E}, \mathit{\Pi})) : \begin{array}{l} (\bar{H}, \mathsf{crs}') \leftarrow_\$ \mathsf{Init}(1^\lambda) \\ \bar{S} \leftarrow [\mathbf{g}, a - h_1, -h_2] \\ \forall i \in [n] : w_i \leftarrow_\$ \mathbb{Z}_q, R_i \leftarrow_\$ \mathbb{G} \\ \bar{E}_i \leftarrow \underline{w_i \cdot \bar{H}} + (0, R_i), \\ \bar{R}_i \leftarrow w_i \cdot ([\mathbf{g}, a, 0] - (0, 0, \bar{H})) \\ \mathit{\Pi}'_i \leftarrow \mathsf{S}_4(\mathsf{td}^s, (\bar{S}, \bar{R}_i)) \end{array} \right\}$$

Claim 24 *If DDH assumption holds over $\mathbb{G}$ chosen from the sequence S then $\mathbf{H}_2 \approx_c \mathbf{H}_1$.*

Proof (Sketch). We can use an hybrid argument to switch one by one the elements $\bar{E}_i$. In each step of the hybrid argument we reduce to the problem of distinghuish the tuple $[\mathbf{d}, \mathbf{f}], w \cdot [\mathbf{d}, \mathbf{f}]$ from $[\mathbf{d}, \mathbf{f}], w \cdot [\mathbf{d}], e \cdot [\mathbf{f}]$ where $[\mathbf{d}] \in \mathbb{G}^2$ and $[\mathbf{f}] \in \mathbb{G}^2$. Easily, the indistinguishability of this problem is implied by DDH assumption. Given $[\mathbf{d}, \mathbf{f}], \bar{Z}_1, \bar{Z}_2$, where $\bar{Z}_1 = Z_{1,1}, Z_{1,2}$, from the challenger we set $[\mathbf{g}] \leftarrow [\bar{d}], \bar{H} \leftarrow [\bar{f}]$, $a \leftarrow \alpha \cdot [g_2]$ for $\alpha \leftarrow_\$ \mathbb{Z}_q$, $\bar{W}_i = \bar{Z}_1, \alpha \cdot Z_{1,2} + (0, 0, R_i)$ and $\bar{E}_i \leftarrow \bar{Z}_2 + (R_i, 0)$.

Claim 25 *If $\mathcal{NIZK}_4$ is adaptive multi-theorem zero-knowledge then $\mathbf{H}_2 \approx_c \mathbf{R}_{\mathcal{NIZK}_{\mathtt{sd}}, \bar{U}_S}$.*

Proof. The difference between $\mathbf{H}_2$ and $\mathbf{R}_{\mathcal{NIZK}_{\mathtt{sd}}, \bar{U}_S}$ is only that former use the simulator of $\mathcal{NIZK}_4$ while the latter use the prover of $\mathcal{NIZK}_4$, the reduction is trivial.

Remark 4 (On the sufficiency of distributional zero-knowledge.). In the proof of the theorem 6, specifically in the proof of the lemmas for the hybrid $\mathbf{H}_4$ and $\mathbf{H}_5$ we need to simulate the proof Π_j^{s}, however the instance $[\mathbf{g}, a], \bar{W}$ are indeed sampled from $\mathcal{U}_{\mathtt{prm}}$, in fact $[\mathbf{g}, a]$ are sampled at initialization and $a = [\mathbf{g}]^T \cdot \mathbf{a}$, so $[a]$ is distributed uniformly at random, and the w, R are sampled uniformly at random, since $j \in [n] \setminus \mathbf{C}^S$.

Theorem 16. *Let f be the function that upon input $(w, R) \in \mathbb{Z}_q \times \mathbb{G}$ outputs R. The proof system $\mathcal{NIZK}_{\mathtt{sd}}$ is f-simulation extractable.*

Proof. We define the extractor Ext to be the function that upon input a proof $(\bar{E}, \Pi)$ and the trapdoor td^e outputs $R \leftarrow E_1 - \mathsf{td}^e \cdot E_2$. Let A be an adversary that plays the simulation extractability experiment for $\mathcal{NIZK}_{\mathtt{sd}}$. In particular the adversary outputs an instance $[\mathbf{g}', a'], \bar{W}'$ and a proof $\bar{E}', \Pi'$ such that the $\mathsf{V}_4(\mathsf{crs}_4, [\mathbf{g}', a' - h_1, -h_2], (\bar{W}', 0) - (0, 0, E'), \Pi') = 1$ but for any w' the following holds $\bar{W} \neq w' \cdot [\mathbf{g}', a'] + (0, 0, \tilde{R})$.

By simulation soundness of $\mathcal{NIZK}_4$, with overwhelming probability, there exists w such that $(\bar{W}', 0) - (0, 0, \bar{E}') = w \cdot [\mathbf{g}', a' - h_1, -h_2]$.

Moreover, by the definition of $\tilde{R}$ there exist a value w'' such that:

$$(\bar{W}', 0) - (0, 0, -w'' \cdot h_1 - \tilde{R}, -w'' \cdot h_2) = w \cdot [\mathbf{g}', a' - h_1, -h_2]$$

In particular, this implies that $w'' = w$. By substitution and simple calculations:

$$(\bar{W}', 0) = w \cdot [\mathbf{g}', a' - h_1, g_2] + w \cdot (0, 0, h_1, h_2) + (0, 0, \tilde{R}, 0) = w \cdot [\mathbf{g}', a', 0] + (0, 0, \tilde{R}, 0)$$

The last equation is in contradiction with our assumption that the adversary A breaks f-simulation extractability.