Publicly Auditable Functional Encryption

Vlasis Koutsos and Dimitrios Papadopoulos

The Hong Kong University of Science and Technology
{vkoutsos,dipapado}@cse.ust.hk

Abstract. We introduce the notion of *publicly auditable functional encryption* (PAFE). Compared to standard functional encryption, PAFE operates in an extended setting that includes an entity called *auditor*, besides key-generating authority, encryptor, and decryptor. The auditor requests function outputs from the decryptor and wishes to check their correctness with respect to the ciphertexts produced by the encryptor, without having access to the *functional secret key* that is used for decryption. This is in contrast with previous approaches for result verifiability and consistency in functional encryption that aim to ensure decryptors about the legitimacy of the results they decrypt.

We propose four different flavors of public auditability with respect to different sets of adversarially controlled parties (only decryptor, encryptor-decryptor, authority-decryptor, and authority-encryptor-decryptor) and provide constructions for building corresponding secure PAFE schemes from standard functional encryption, commitment schemes, and non-interactive witness-indistinguishable proof systems. At the core of our constructions lies the notion of a *functional public key*, that works as the public analog of the functional secret key of functional encryption and is used for verification purposes by the auditor. Crucially, in order to ensure that these new keys cannot be used to infer additional information about plaintext values (besides the requested decryptions by the auditor), we propose a new indistinguishability-based security definition for PAFE to accommodate not only functional secret key queries (as in standard functional encryption) but also functional public key and decryption queries. Finally, we propose a publicly auditable multi-input functional encryption scheme (MIFE) that supports inner-product functionalities and is secure against adversarial decryptors. Instantiated with existing MIFE using "El Gamal"-like ciphertexts and Σ-protocols, this gives a lightweight publicly auditable scheme.

Keywords: Functional Encryption, Auditability, Public Verifiability

1 Introduction

Functional Encryption (FE) [16,30] is a cryptographic primitive that transcends the "all-or-nothing" decryption capabilities that classical public-key encryption schemes provide, by allowing the decryptor to acquire pre-agreed function outputs of the encryptor's data. In particular, given an *encryption key* ek, the encryptor produces a ciphertext ct for plaintext data x. The decryptor can then use a *functional secret key* sk_f, for an agreed function f, provided by a key-generation authority, in order to retrieve the function output $f(\mathsf{x})$. The security of FE guarantees that the decryptor learns *nothing* about x other than $f(\mathsf{x})$ (and what can be inferred from it). Since it was first proposed by Boneh et al. [16] there has been a plethora of works on FE e.g., providing efficient schemes for specific functionalities (inner product [7], quadratic [5,8,11,24]), and generalizations of the definition for multiple inputs (Multi-Input FE (MIFE) [23,35]), multiple encryptors or authorities (Multi-Client FE (MCFE) [4,20,28] and Multi-Party FE [6]), or dynamic sets of participants (Decentralized Dynamic FE [19]).

The selective decryption capabilities that FE provides makes it suitable for multiple real-world applications, such as the following:

Checkable cloud storage. Consider an application where users wish to upload their files (e.g., images) onto a cloud service provider. For privacy purposes uploaded files should be encrypted. However, from the cloud service provider's perspective it would be ideal if it could still infer some information about the encrypted files, with the consent of the users. For example, it might wish to learn certain file-related aggregate statistics, or ensure that the files' contents *satisfy certain policies* (e.g., the uploaded images do not contain states of déshabillé [2]). Using FE to produce the ciphertext and providing the cloud server with appropriate sk_f simultaneously achieves the above privacy and utility properties.

Auditable financial data. Another example application comes from the field of finance, where there exist institutions (e.g., [1,3]) that are in charge of examining the books of business-conducting companies and organizations to ensure that they conform to legal and financial policies. Again, FE enables such fine-grained control without necessarily disclosing all raw data.

Private data brokerage. The abundance of data produced daily from our online activity has naturally led to the notion of "data monetization". Companies operating as *data brokers* collect large volumes of data, perform useful statistical analyses over them and market the results to interested buyers (e.g., for marketing or political campaigns [21]). Due to the potentially sensitive nature of the raw data, privacy-aware individuals could opt to use an FE scheme that only reveals to the broker pre-agreed aggregate statistics.

While FE seems to map very nicely to the privacy requirements of these applications, the trust model in the settings described allows for potential misbehavior from parties that is not captured by FE security definitions. Note that in all applications above *the party that plays the role of the FE-decryptor is different than the party that needs to be convinced about the legitimacy of the function outputs.* For example, the cloud service provider may need to prove to a law enforcement agency that the stored encrypted data do not violate any legal policies and likewise, for our second example, auditing companies may need to provide guarantees to governmental regulatory bodies. Finally, in our last example, data buyers should receive a guarantee about the correctness and consistency of the purchased statistical analyses results with respect to the raw data, to avoid having to blindly trust the broker. Based on the above, FE only partially achieves the security requirements of the mentioned applications (allowing the decryptor to only learn function outputs). Particularly, it needs to be augmented or complemented with other cryptographic techniques to enable results verification.

There exists a limited number of works in the FE literature that aims to enrich schemes with some notion of result verification. Badrinarayanan et al. [10] proposed *verifiable FE* and more recently Badertscher et al. [9] proposed a relaxation of this notion under various adversarial models called *consistent FE*. However, both these works operate in a different security model than the ones described above, as they *aim to protect the decryptor* against attacks launched by a misbehaving encryptor, authority, or both. That is, at a very high level, the decryptor can run a verification algorithm that ensures there is a common "explanation" for all decrypted function outputs (see detailed discussion in related work section below). That said, this verification algorithm may require access to the functional decryption keys sk_f for different functions f, which is perfectly acceptable in case it will be executed by the decryptor (who is already entrusted with these keys). Importantly, the problem we are concerned with is strictly harder: an external party needs to be able to verify decrypted results without access to sk_f and *the decryptor is always assumed to behave adversarially to this goal*. Hence none of these prior works for verifiable FE is applicable.

One "easy" solution to our problem would be to provide the external auditing party with sk_f, allowing it to decrypt results directly. However, access to sk_f allows the decryption of arbitrary sets of ciphertexts (e.g., in an application where ciphertexts are publicly accessible). While this might be acceptable for some cases, a more "controlled disclosure," is more realistic and preferable in other scenarios i.e., when the auditor has to go through the decryptor first. Finally, an alternative "folklore" way for verifying received results is if one can rely on an active direct communication channel between the encryptor and the auditor. In such a setting, the encryptor can use a generic zero-knowledge proof (ZKP) protocol to convince the auditor, when asked, that its plaintext does not deviate from certain policies. This is far from a realistic assumption though, as the encryptor might be unavailable or even unwilling to provide the auditor with such information. An alternative to this would be for the encryptor to produce, ahead of time and alongside its ciphertext, corresponding proofs of validity, provably adhering to certain policies. Policies change, unfortunately, meaning that the encryptor would need to engage in multiple ZKP executions over time, which is undesirable for the same reasons as the previous example.

1.1 Our results

Motivated by the above, we introduce the notion of *publicly auditable functional encryption (PAFE)*, which operates in an extended setting that includes, besides the authority, encryptor, and decryptor, an additional party named *auditor*. PAFE enables the auditor to verify the decryption results/functional outputs received from the decryptor. Unlike previous works on FE verification [9, 10], PAFE achieves "public verifiability", i.e., the auditor can run the verification algorithm *without access to* sk_f. In order to achieve this, we introduce a new

Sec	Encryptor	Authority	Decryptor	Public Auditability	Security	Building Blocks	
5.1	○	○	●	PA-UD	Ad	Ad-FE	Com + NIWI
5.1	●	○	●	PA-UED	Ad	Ad-FE	
5.2	○	●	●	PA-UAD	Ad	4×Ad-FE	
5.2	●	●	●	PA-UEAD	Sel	4×Sel-FE	

Table 1. Auditability versions for general FE depending on the untrusted parties and cryptographic building blocks used for the corresponding PAFE schemes. Legend: ○=Honest, ●=Adversarial. Ad/Sel stands for adaptively/selectively secure.

public parameter that we term *functional public key* (pk_f) for function f. This is provided by the authority as a public analog of sk_f and it operates as an "anchor-of-trust" for the auditor's verification.

Next, we define different flavors of public auditability, based on different corrupted sets among authority, encryptor, and decryptor. We stress that public auditability is meaningful as a property when the decryptor is untrusted; it is trivially achieved if the auditor trusts the decryptor-provided results. Besides, any combination of the remaining entities (encryptor, authority, or both) may be colluding with the decryptor to "fool" the auditor. Thus, we propose four definitions of auditability for FE corresponding to different untrusted-participants sets PA-UD, PA-UED, PA-UAD, PA-UEAD corresponding to untrusted decryptor, encryptor-decryptor, authority-decryptor, and encryptor-authority-decryptor, respectively, as shown in Table 1, and we explore the relations among them.

Besides auditability, PAFE must maintain the standard security definition of FE [16, 20] i.e., the decryptor learns nothing about the plaintext, except for function outputs. Additionally, the inclusion of functional public keys enables a broader class of attacks from an adversary that has access *only* to ciphertexts, pk_f, and possibly function outputs y; but not sk_f. To elevate the security model to a more realistic scenario that includes the cases we explained above, we consider a mixed class of adversaries that may acquire access to functional secret or public keys *in a dynamic way*. Additionally, we provide the adversary with oracle access to the encryption and decryption algorithms. Recall that a secure FE scheme allows for any adversary to win with non-negligible advantage only if for all queried functional secret keys and ciphertexts, all underlying plaintexts pairs have equal function outputs. In our PAFE extended model, adversaries may attempt to abuse the decryption capabilities on a ciphertext ct to infer functional evaluations for a function f for which acquiring the respective sk_f would violate the FE winning conditions. To capture all adversarial cases, we propose a new security definition for PAFE, extending the one for FE.

Additionally, we present four different constructions, each one satisfying a different public auditability flavors. Table 1 depicts the respective cryptographic components and the achieved Security/Public-Auditability types. As we discuss in the next part, we build our PAFE using secure FE schemes, enhanced with appropriate use of commitment schemes to "anchor" the functional public keys pk_f, and non-interactive witness indistinguishability (NIWI) proof systems to force the untrusted parties to prove the correctness of their computations. For the untrusted key-generating authority case (PA-UAD, PA-UEAD), we also expand upon the replication techniques of Badrinarayanan et al. [10].

Our first four schemes work for arbitrary classes of functions (as long as the underlying FE supports them). In the last part of the paper we design a publicly auditable multi-input FE (MIFE) scheme specifically for inner-product functionalities, i.e., linear combinations between encrypted vectors and public weights. Using the MIFE of [7], which produces "El Gamal"-style ciphertexts as a building block, combined with classic Σ-protocols [13,17] yields an efficient publicly auditable scheme against untrusted decryptors.

1.2 Overview of techniques

The introduction of the functional public key in the FE setting poses a "modeling" challenge. Even though pk_f should be uniquely tied to sk_f for a specific function f, it should not reveal anything about its private counterpart or the plaintext of the encryptor. To ensure this, we compute pk_f as a perfectly binding and computationally hiding commitment of sk_f. Due to the perfectly binding property no adversary can generate two different functional secret keys that map to the same functional public key which makes the auditability properties easier to prove. The computational hiding property ensures that no bounded-resources adversary can infer any information about sk_f from its pk_f counterpart, hence function outputs that have not been explicitly queried for via the decryption oracle are protected.

However, augmenting IND-secure FE schemes to build secure PAFE schemes is not trivial, due to the diversification of the winning conditions between the IND-security game for FE schemes and that of PAFE that we mentioned above. One way to achieve this would be to restrict ourselves to weaker adversaries. E.g., consider an extremely limited setting where the adversary (after setup) first declares all the secret and public functional keys it wants, as well as all plaintext pairs to be encrypted and ciphertexts to be decrypted. Having access to all this information allows us to check which winning conditions are not violated each time and construct the keys accordingly (either "honestly" or as "dummies"). Clearly, we would prefer to achieve security against more general adversarial behavior. To this end, we use NIWI proofs to combat such adversarial behavior based on the different winning conditions of the two games and achieve fully *adaptive* security, i.e., there is no restriction in the order in which the adversary issues its different queries. Specifically, our PA-UD and PA-UED constructions leverage the setup performed by the trusted authority to construct an adaptively secure PAFE scheme based on an adaptively secure FE scheme, perfectly binding and computationally hiding commitments and perfectly sound and computationally witness indistinguishable NIWI proof system.

For the cases where the authority is considered untrusted, we need to employ multiple instances of FE where the function output derives from the *majority* of the decryptions and enrich our NIWI relations accordingly, using techniques similar to [10]. Importantly, in the $\mathsf{PA\text{-}UAD}$ setting we can still construct an adaptively secure PAFE scheme by leveraging that the encryptor is trusted. This derives specifically from a combination of all ciphertexts using a common plaintext x with complex relations proving a threshold of correct execution of the Setup, Key-Generation, and Decryption algorithms. However, in the last case where all entities are untrusted $\mathsf{PA\text{-}UEAD}$ we do not have such guarantees and we can only achieve a *selective* type of security, as follows: We require that the encryptor provide all plaintext pair queries, before generating any functional secret keys. While this is more limited than the adaptive security of the first three schemes, it arguably suffices for certain applications (e.g., for the cloud application we discussed above, consider the case where all data is uploaded ahead of time, before auditing functions are chosen).

The four different corruption sets result in four discrete public auditability definitions of their own, which we divide into two subcategories depending on whether the authority is trusted or not. A similar division can also be done on whether the encryptor is trusted or not, which results in an interesting realization about the public auditability definitions. For the former cases ($\mathsf{PA\text{-}UD}$,$\mathsf{PA\text{-}UAD}$), the auditor is guaranteed to receive the function output $f(\mathsf{x})$ that corresponds to the plaintext x that the encryptor has used to produce its ciphertext, and specifically for function f that the auditor holds its respective pk_f. However, for the cases where the encryptor is untrusted ($\mathsf{PA\text{-}UED}$,$\mathsf{PA\text{-}UEAD}$), it can generate its ciphertext arbitrarily. Therefore, a notion of "consistency" has to suffice to the auditor, which results to an existential condition for x to be in the domain of f $\big($whose respective pk_f the auditor holds and decryption returns $f(\mathsf{x})\big)$.

2 Related work

Badrinarayanan et al. [10] were the first to consider, in the FE setting, the possibility of encryptors colluding with the authority to try and "cheat" the decryptor into receiving falsified or meaningless function outputs. To safeguard against such attacks, they proposed verifiable FE, including algorithms for the verification of the ciphertext production and the key generation. The verifiability property states that if both algorithms succeed, then decryptors always get a function output (for function f) of a plaintext in the domain of f. This is a very strong definition, as it quantifies over all mpk, ct, f, sk_f the existence of a plaintext x whose function outputs are the result of the decryption algorithm. Interestingly, this is actually the best decryptors can ask for, since the ciphertext can be generated arbitrarily. More efficient verifiable FE schemes were more recently proposed for inner-product functionalities using pairing-based NIWIs and a perfectly correct inner-product functional encryption scheme [33], and for general functionalities via the use of trusted execution environments [34].

Following in the same line of works, Badertscher et al. [9] proposed the notion of consistency for FE schemes which builds on the idea of [10] in the following way. Additionally to considering the case where both the encryptor and the authority collude, they examine also the cases where just one of the two entities try to "cheat" the decryptor. They observe that in both cases where the encryptor is corrupted, a similar property as in [10] suffices, however, when the encryptor is honest, then the decryptor could request a stronger guarantee, i.e., that it gets the function output of the *specific* plaintext x that the encryptor used to generate the ciphertext ct. The authors explore the relations between their consistency notions and other notions of security (i.e., IND-CPA, IND-CCA, CFE), provide compilers to build consistent FE schemes from FE, NIZKs, and NIWIs, and provide concrete constructions for consistent FE for inner-product functionalities, in the presence of a corrupted encryptor.

At first, it might seem that our notion of public auditability can be achieved by the above schemes. However, even though there exist similarities specifically between our constructions and the ones of [9, 10], public auditability aims to protect the auditor who lacks knowledge of sk_f and knows instead only pk_f. In that sense, our approach can be broadly viewed as a public-key analog of the notions of verifiability and consistency. Thus, not only our constructions require additional techniques for auditability, but PAFE requires a modified security definition, expanded to capture additional adversarial cases, as discussed above.

Koutsos et al. [27] proposed a construction of a privacy-preserving data marketplace that uses FE to protect the privacy of the raw data. To the best of our knowledge, this is the only work that considers a similarly augmented setting for FE with auditability and provides a solution without relying on trusted communication between the encryptor and the auditor. Similarly to our work, they utilize a public equivalent of sk_f to ensure the legitimacy of the brokered results against an untrusted auditor (but not untrusted authority/encryptor or any combination between them). Moreover, they only consider "passive" attacks from parties that observe ciphertexts but without decryption capabilities. Finally, their scheme builds on the MCFE scheme of [20] which supports only inner-products functionalities.

Barbosa et al. [12] were concerned with a somewhat similar problem to ours. In that work the authors proposed a cryptographic primitive called Delegatable Homomorphic Encryption (DHE). With DHE schemes a weak client is convinced that the decryption (which was performed by a potentially malicious cloud service provider) was performed correctly. The authors assume a trusted communication channel between the encryptor and the auditor, so that the latter could be convinced about the legitimacy of the results received from decryptors. However, this is a strong assumption and an unrealistic one as well, as the content provider could go offline after the uploading of its data onto the cloud service provider. Contrary to our work, the authors of [12] do not consider any possible corruptions from the encryptor or the authority and rely on both FE and fully-homomorphic encryption [22] schemes.

3 Preliminaries

Below we present the necessary cryptographic background for our work.

```
$\mathsf{WI}_{\beta}^{\mathsf{NIWI}}(1^{\lambda}, \mathcal{A})$ (for relation R)
$(x, w_1, w_2, st) \leftarrow \mathcal{A}_1(1^{\lambda})$
$\pi \leftarrow \mathsf{NIWI.Prove}(1^{\lambda}, x, w_{\beta})$
$\alpha \leftarrow \mathcal{A}_2(\pi, st)$
**Output:** $\alpha \wedge (x, w_1) \in R \wedge (x, w_2) \in R$
```

Fig. 1. Witness-indistinguishability game of a NIWI proof system.

General Notation. We denote by $\mathcal{F} = \{\mathcal{F}_{\lambda}\}_{\lambda \in \mathbb{N}}$ a family of sets $\mathcal{F}_{\lambda}$ of functions $f : \mathcal{X}_{\lambda} \rightarrow \mathcal{Y}_{\lambda}$. We call $\mathcal{F}_{\lambda}$ a functionality class such that all functions $f \in \mathcal{F}_{\lambda}$ have the same domain and the same range. A function $negl(\lambda)$: $\mathbb{N} \leftarrow \mathbb{R}^{+}$ is negligible if for every positive polynomial $p(\lambda)$ there exists a $\lambda_0 \in \mathbb{N}$, such that for all $\lambda > \lambda_0 : \epsilon(\lambda) < 1/p(\lambda)$. We denote $[n] = \{1, \cdots, n\}$ for $n \in \mathbb{N}^{\star}$. For algorithms $\mathcal{A}$ and $\mathcal{B}$, we write $\mathcal{A}^{\mathcal{B}(\cdot)}(x)$ to denote that $\mathcal{A}$ gets x as an input and has oracle access to $\mathcal{B}$, that is, the response for an oracle query q is $\mathcal{B}(q)$. Oracles increment a counter every time they receive a query and associate the input-output pairs with their counter, so they can answer repeated queries consistently. Last, we denote by $\leftarrow\!\!\$\ D$ sampling uniformly at random from domain D.

NIWI Proof Systems. A *non-interactive witness-indistinguishable proof system (NIWI)* allows a prover to convince a verifier about the validity of the statement in a way that guarantees that "cheating" provers cannot succeed in this. On the other hand, witness indistinguishability means that no verifier interacting with an honest prover can distinguish which of two witnesses w_1, w_2 was used by the latter (assuming such witnesses exist for the statement).

Definition 1. (Non-Interactive Witness-Indistinguishable Proofs [9]) *Let R be an NP relation and consider the language $L = \{x \mid \exists\ w \text{ with } (x, w) \in R\}$. A non-interactive witness-indistinguishable proof (NIWI) for the relation R is a tuple of PPT algorithms* $\mathsf{NIWI} = (\mathsf{NIWI.Prove}, \mathsf{NIWI.Verify})$:
$\mathsf{NIWI.Prove}(1^{\lambda}, x, w)$: *Takes as input the unary representation of the security parameter λ, a statement x and a witness w, and outputs a proof π.*
$\mathsf{NIWI.Verify}(1^{\lambda}, x, \pi)$: *Takes as input the unary representation of the security parameter λ, a statement x, and a proof π, and outputs* 0 *or* 1.

A NIWI is complete, if for all statement-witness pairs in the relation $(x, w) \in R$, it holds that: $\Pr[\mathit{NIWI.Verify}(1^{\lambda}, x, \mathit{NIWI.Prove}(1^{\lambda}, x, w)) = 1] = 1$. Besides, a NIWI fulfills the properties of soundness and witness-indistinguishability.

(NIWI Soundness). We define the advantage of an adversary $\mathcal{A}$ as:

$$Adv_{\mathit{NIWI},\mathcal{A}}^{Sound}(\lambda) \coloneqq \Pr[(x, w) \leftarrow \mathcal{A}(1^{\lambda}) : \mathit{NIWI.Verify}(x, \pi) = 1 \wedge x \notin L]$$

A NIWI is perfectly sound if $Adv_{\mathit{NIWI},\mathcal{A}}^{Sound}(\lambda) = 0$ for all algorithms $\mathcal{A}$, and computationally sound, if $Adv_{\mathit{NIWI},\mathcal{A}}^{Sound}(\lambda) \leq \mathsf{negl}(\lambda)$ for all PPT algorithms $\mathcal{A}$.

(Witness-Indistinguishability). For $\beta \in \{0, 1\}$, we define the experiment $\mathit{WI}_{\beta}^{NIWI}(1^{\lambda}, \mathcal{A})$ in Figure 1. The advantage $Adv_{NIWI}^{WI}(\mathcal{A}(1^{\lambda}))$ of an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is:

$$|\Pr[\mathsf{WI}_0^{\mathsf{NIWI}}(1^\lambda,\mathcal{A})=1]-\Pr[\mathsf{WI}_1^{\mathsf{NIWI}}(1^\lambda,\mathcal{A})=1]|$$

A NIWI is witness-indistinguishable, if $Adv^{WI}_{NIWI}\big(\mathcal{A}(1^\lambda)\big)=0$ for all algorithms $\mathcal{A}=(\mathcal{A}_1,\mathcal{A}_2)$, and computationally witness-indistinguishable, if the advantage $Adv^{WI}_{NIWI}\big(\mathcal{A}(1^\lambda)\big)\leq \mathsf{negl}(\lambda)$ for all PPT algorithms $\mathcal{A}=(\mathcal{A}_1,\mathcal{A}_2)$. In our PAFE schemes, we use NIWIs with perfect soundness and computational witness indistinguishability, such as the ones proposed in [14]. We also note that for the configurations with trusted PAFE authority (Section 5.1), our NIWIs can be readily replaced with zero-knowledge proof systems that provide a stronger flavor of witness privacy.

Commitments. This primitive allows a party to first commit to a message in a way that reveals nothing about it, and later open it in way that guarantees it cannot "equivocate" for multiple openings. We rely on perfectly binding, computationally hiding commitments (e.g., built from one-way permutations [14]).

Definition 2 (Commitment Schemes). *A commitment scheme consists of a pair of PPT algorithms (Com.Setup,Com.Commit). The Com.Setup algorithm pp $\leftarrow$ Com.Setup(1^λ) generates public parameters pp for the scheme, for security parameter λ. The commitment algorithm defines a function $\mathcal{M}_{pp}\times R_{pp}\rightarrow \mathcal{C}_{pp}$, for message space $\mathcal{M}_{pp}$, for randomness space R_{pp}, and for commitment space $\mathcal{C}_{pp}$, determined by pp. It takes as input a message x and randomness r and outputs $c\leftarrow \mathsf{Com.Commit}_{pp}(x;r)$. A perfectly binding and computationally hiding commitment scheme must satisfy the following properties:*

- ***Perfectly Binding:*** *Two different strings cannot have the same commitment. More formally, $\forall\ x_0\neq x_1, r_0, r_1, \mathsf{Com.Commit}(x_0;r_0)\neq \mathsf{Com.Commit}(x_1;r_1)$.*
- ***Computationally Hiding:*** *For all strings x_0 and x_1 (of the same length), for all* non-uniform *PPT adversaries $\mathcal{A}$, we have that $Adv^{Com-Hiding}(\mathcal{A})$ is the advantage defined as:*
  $|\Pr[\mathcal{A}(\mathsf{Com.Commit}(x_0;r_0))=1]-\Pr[\mathcal{A}(\mathsf{Com.Commit}(x_1;r_1))=1]| < \mathsf{negl}(\lambda)$

3.1 Functional encryption

A functional encryption scheme [8,16] can be used to encrypt a message, akin to "standard" encryption. However, using special function-specific decryption keys, it also allows a decryptor to learn specific function outputs of the message, but nothing else. The following definition is based on [16,20].

Definition 3 (Functional Encryption). *Let $\mathcal{F}=\{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of sets $\mathcal{F}_\lambda$ of functions $f:\mathcal{X}_\lambda\rightarrow\mathcal{Y}_\lambda$. An FE scheme for the functionality class $\mathcal{F}_\lambda$ is a tuple of four algorithms* FE = (FE.Setup, FE.KeyGen, FE.Enc, FE.Dec).

- FE.Setup(1^λ): *Takes as input the security parameter λ as 1^λ and outputs a master public key* mpk*, a master secret key* msk*, and an encryption key* ek.
- FE.KeyGen(mpk,msk,f): *Takes as input a master public key* mpk*, a master secret key* msk *and a function $f\in\mathcal{F}_\lambda$, and outputs a functional secret key* sk_f.

– $\mathsf{FE.Enc}(\mathsf{ek},\mathsf{x})$: *Takes as input an encryption key* ek *and a string* $\mathsf{x} \in \mathcal{X}_\lambda$*, and outputs a ciphertext* ct*.*
– $\mathsf{FE.Dec}(\mathsf{mpk},f,\mathsf{sk}_f,\mathsf{ct})$: *Takes as input a master public key mpk, a function* $f \in \mathcal{F}_\lambda$*, a functional secret key* sk_f*, and a ciphertext* ct*. It outputs a function value* $\mathsf{y} \in \mathcal{Y}_\lambda$ *or* $\perp$*, indicating an invalid ciphertext.*

Correctness. *An FE scheme is correct, if* $\forall \lambda \in \mathbb{N}$, $\forall f \in \mathcal{F}_\lambda, \exists x \in \mathcal{X}_\lambda$ *s.t.:*

$$\Pr\left[\mathsf{FE.Dec}(\mathsf{mpk},f,\mathsf{sk}_f,\mathsf{ct}) = f(x) \middle| \begin{array}{l}(\mathsf{msk},\mathsf{mpk},\mathsf{ek}) \leftarrow \mathsf{FE.Setup}(1^\lambda)\\ \mathsf{sk}_f \leftarrow \mathsf{FE.KeyGen}(\mathsf{mpk},\mathsf{msk},\ f)\\ \mathsf{ct} \leftarrow \mathsf{FE.Enc}(\mathsf{ek},\mathsf{x})\end{array}\right] > 1 - negl(\lambda)$$

The security of FE is captured by the following indistinguishability game. At a high level, the adversary has access to a "left-right" encryption oracle and a key generation for functions of its choice. Importantly, the game detects whether the adversary ever submit an encryption query for a pair of messages with different output for some of the queried functionalities (trivially breaking the game).

Definition 4 (IND Security for FE). *Let us consider an FE scheme. No PPT adversary* $\mathcal{A}$ *should be able to win the following game against a challenger* $\mathcal{C}$*:*

- *Initialization:* $\mathcal{C}$ *runs the setup algorithm* $(\mathsf{mpk},\mathsf{msk},\mathsf{ek}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$ *and chooses a random bit* $b \leftarrow\$ \{0,1\}$*. It provides the master public key* mpk *to* $\mathcal{A}$*.*
- *Encryption queries* $\mathrm{QEnc}(\mathsf{x}_0,\mathsf{x}_1)$*:* $\mathcal{A}$ *has unlimited adaptive access to a Left-or-Right encryption oracle and receives ciphertext* ct*, generated by* $\mathsf{FE.Enc}(\mathsf{ek},\mathsf{x}_b)$*.*
- *Functional key queries* $\mathrm{QKeyGen}(f)$*:* $\mathcal{A}$ *has unlimited and adaptive oracle access to the* $\mathsf{FE.KeyGen}(\mathsf{mpk},\mathsf{msk},f)$ *algorithm for any input function* f *of its choice. It is given back the functional secret key* sk_f*.*
- *Finalize:* $\mathcal{A}$ *provides its guess* b' *on the bit* b *and this procedure outputs the result* β *of the security game, according to the analysis given below.*

The game output β *depends on the following conditions. If* QKeyGen *queries have been issued for some function* f *and there exists a pair of values* $(\mathsf{x}_0,\mathsf{x}_1)$ *queried to* QEnc*, such that* $f(\mathsf{x}_0) \neq f(\mathsf{x}_1)$*, we set* $\beta \leftarrow\$ \{0,1\}$*. In any aother case we set the output to* $\beta \leftarrow b'$*.*
$\mathcal{A}$ *wins in the game if* $\beta = b$ *and we remark that a naive adversary, by sampling randomly* β *has probability of winning equal to* $\frac{1}{2}$*. We denote the advantage that* $\mathcal{A}$ *has of winning as* $Adv^{IND}(\mathcal{A})$ *and we say this FE is* IND-secure *if for any PPT adversary* $\mathcal{A}$*,* $Adv^{IND}(\mathcal{A}) = |Pr[\beta = 1|b = 1] - Pr[\beta = 1|b = 0]| \leq negl(\lambda)$*.*

The game above captures the adaptive security of FE. There exists also a selective variant, where the adversary is forced to send all its encryption queries QEnc in one shot, before issuing any other type of query.

3.2 Multi Input Functional Encryption

We alter the original definition of MIFE scheme in [23] for consistency purposes. Specifically, we let $\mathsf{MIFE.Setup}(\cdot)$ output additionally a master public key mpk and we augment all remaining algorithms' inputs with mpk.

```
**Experiment** $\mathsf{IND}_{\mathcal{A}}^{\mathsf{MIFE}}(1^\lambda)$
    $(n, \mathrm{st}_0) \leftarrow \mathcal{A}_0(1^\lambda)$
    $\{\mathsf{ek}_i\}_{i\in[n]} \leftarrow \mathsf{MIFE.Setup}(1^\lambda, n)$
    $(\vec{X}^0, \vec{X}^1, st_1) \leftarrow \mathcal{A}_1^{\mathsf{MIFE.KeyGen(mpk,msk,\cdot)}}(st_0, \{\mathsf{ek}_i\}_{i\in[n]}),\ \vec{X}^\ell = \{x_{i,j}^\ell\}_{i\in[n],j\in[q]}$
    $b \leftarrow \{0,1\}\ \mathsf{CT}_{i,j} \leftarrow \mathsf{MIFE.Enc}(\mathsf{mpk}, \mathsf{ek}, \vec{X})$
```

Fig. 2. MIFE security game.

Definition 5 (Multi-Input Functional Encryption). *Let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of sets $\mathcal{F}_\lambda$ of functions $f : \mathcal{X}_\lambda \to \mathcal{Y}_\lambda$. A multi-input functional encryption scheme (MIFE) for the functionality class $\mathcal{F}_\lambda$ is a tuple of four algorithms* MIFE= (MIFE.Setup, MIFE.KeyGen, MIFE.Enc, MIFE.Dec)*.*

MIFE.Setup$(1^\lambda, n)$: *Takes as input a unary representation of the security parameter λ and an integer n, and outputs a master public key* mpk*, a master secret key* msk*, and a set of n encryption keys $\{\mathsf{ek}_i\}_{i\in[n]}$.*

MIFE.KeyGen(mpk,msk,f): *Takes as input a master public key* mpk*, a master secret key* msk *and a function $f \in \mathcal{F}_\lambda$, and outputs a functional secret key sk_f.*

MIFE.Enc(mpk,ek_i,x): *Takes as input a master public key* mpk*, an encryption key ek_i and a string $x \in \mathcal{X}_\lambda$, and outputs a ciphertext ct_i or* err *(denoting an encryption error).*

MIFE.Dec(mpk,f,sk_f,$\{\mathsf{ct}_i\}_{i\in[n]}$): *Takes as input a master publi c key pk_f, a function f, a functional key sk_f, and a set of ciphertexts $\{\mathsf{ct}_i\}_{i\in[n]}$ and outputs a function value $y \in \mathcal{Y}_\lambda$ or $\perp$, which indicates an invalid ciphertext.*

Security of MIFE schemes was defined in [23], parameterized by the number of encryption keys known to the adversary, and the number of challenge messages per encryption key. Similarly to [25], our construction is uplifted of any conditions regarding these two parameters.

Definition 6 (Indistinguishability-based security [25]). *We say that a MIFE scheme for $n-$ary functions $\mathcal{F}$ is fully IND-secure if for every adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ defined as: $Adv^{sec-MIFE}\big(\mathcal{A}(1^\lambda)\big) = |\Pr[\mathit{IND}_{\mathcal{A}}^{\mathit{MIFE}}] - 1/2| < negl(\lambda)$, where the game $\mathit{IND}_{\mathcal{A}}^{\mathit{MIFE}}(1^\lambda)$ is depicted in Figure 2.*

4 Publicly Auditable Functional Encryption

In this section, we present our definition of publicly auditable functional encryption, its security game, and different flavors of pubic auditability, each one corresponding to a different corruption set among the encryptor, authority, and decryptor entities.

Below we show the algorithms of a PAFE scheme. The main differences compared to FE are: (i) the addition of a functional public key pk_f for each function, (ii) a new algorithm *ProveDec* executed by the decryptor to convince a third party (auditor) about the decryption correctness, and (iii) a new algorithm *PAFE.VerifyDec* that takes pk_f (but not sk_f) and a proof of decryption correctness for the same function f as the key and accepts or rejects an output.

Definition 7 (Publicly Auditable FE). *Let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of sets $\mathcal{F}_\lambda$ of functions $f : \mathcal{X}_\lambda \to \mathcal{Y}_\lambda$. A publicly auditable functional encryption scheme for the functionality class $\mathcal{F}_\lambda$ is a tuple of six algorithms* $\mathsf{PAFE} = (\mathsf{PAFE.Setup}$, $\mathsf{PAFE.KeyGen}$, $\mathsf{PAFE.Enc}$, $\mathsf{PAFE.Dec}$, $\mathsf{PAFE.ProveDec}$, $\mathsf{PAFE.VerifyDec})$.

- $\mathsf{PAFE.Setup}(1^\lambda)$: *Takes as input a unary representation of the security parameter λ and outputs a master public key* mpk*, a master secret key* msk*, and an encryption key* ek*.*
- $\mathsf{PAFE.KeyGen}(\mathsf{mpk},\mathsf{msk},f)$: *Takes as input a master public key* mpk*, a master secret key* msk *and a function $f \in \mathcal{F}_\lambda$, and outputs a functional secret key* sk_f *and a functional pubic key* pk_f*.*
- $\mathsf{PAFE.Enc}(\mathsf{mpk},\mathsf{ek},\mathsf{x})$: *Takes as input an encryption key* ek *and a string $\mathsf{x} \in \mathcal{X}_\lambda$, and outputs a ciphertext* ct*.*
- $\mathsf{PAFE.Dec}(\mathsf{mpk},f,\mathsf{sk}_f,\mathsf{ct})$: *Takes as input a master public key* mpk*, a function $f \in \mathcal{F}_\lambda$, a functional key* sk_f*, and a ciphertext* ct*. It outputs a function value $\mathsf{y} \in \mathcal{Y}_\lambda$ or $\perp$, which indicates an invalid ciphertext.*
- $\mathsf{PAFE.ProveDec}(\mathsf{mpk},f,\mathsf{sk}_f,\mathsf{pk}_f,\mathsf{ct},\mathsf{y})$: *Takes as input a master public key* mpk*, a function $f \in \mathcal{F}_\lambda$, a functional secret key* sk_f*, a functional public key* pk_f*, a ciphertext* ct*, and a value y and outputs a proof π_d.*
- $\mathsf{PAFE.VerifyDec}(\mathsf{mpk},f,\mathsf{pk}_f,\mathsf{ct},\mathsf{y},\pi)$: *Takes as input a master public key* mpk*, a function $f \in \mathcal{F}_\lambda$, a functional public key* pk_f*, a ciphertext* ct*, a value y, and a proof π and outputs 1 if $\mathsf{y} = \mathsf{PAFE.Dec}(\mathsf{mpk},f,\mathsf{sk}_f,\mathsf{ct})$, and 0 otherwise.*

PAFE *satisfies the correctness of FE schemes as per Definition 3 and additionally satisfies the following notion of* auditable correctness*.*

Auditable correctness. *A scheme* PAFE *has auditable correctness, if $\forall\ \lambda \in \mathbb{N}$, $\forall\ f \in \mathcal{F}_\lambda, \exists\ x \in \mathcal{X}_\lambda$ the following probability is negligible in λ.*

$$\Pr\left[\begin{array}{r|l} & (\mathsf{msk},\mathsf{mpk},\mathsf{ek}) \leftarrow \mathsf{PAFE.Setup}(1^\lambda) \\ \mathsf{PAFE.Dec}(\mathsf{mpk},f,\mathsf{sk}_f,\mathsf{ct}) = f(x) & \mathsf{ct} \leftarrow \mathsf{PAFE.Enc}(\mathsf{ek},\mathsf{x}) \\ 1 \leftarrow \mathsf{PAFE.VerifyDec}(\mathsf{mpk},f, & (\mathsf{sk}_f,\mathsf{pk}_f) \leftarrow \mathsf{KeyGen}(\mathsf{mpk},\mathsf{msk},\ f) \\ \mathsf{pk}_f,\mathsf{ct},f(\mathsf{x}),\pi_d) & \pi_d \leftarrow \mathsf{PAFE.ProveDec}(\mathsf{mpk},f,\mathsf{sk}_f,\mathsf{pk}_f, \\ & \qquad \mathsf{ct},f(\mathsf{x})) \end{array}\right]$$

The security of a PAFE scheme is captured by an indistinguishability game that is an "extended" version of the one from Definition 4 to allow also for oracle access to functional public keys and decryptions.

Definition 8 (Security for PAFE). *Consider the following game between PPT adversary $\mathcal{A}$ and challenger $\mathcal{C}$:*

- *Initialization: $\mathcal{C}$ runs the setup algorithm $(\mathsf{mpk},\mathsf{msk},\mathsf{ek}) \leftarrow \mathsf{PAFE.Setup}(1^\lambda)$ and chooses a random bit $b \leftarrow\!\!\$\ \{0,1\}$. It provides the master public key* mpk *to $\mathcal{A}$.*
- *Encryption queries:* $\mathrm{QEnc}(\mathsf{x}_0,\mathsf{x}_1)$*: $\mathcal{A}$ has unlimited adaptive access to a Left-or-Right encryption oracle and receives ciphertext $\mathsf{ct}^b \leftarrow \mathsf{PAFE.Enc}(\mathsf{mpk},\mathsf{ek},\mathsf{x}_b)$.*

```
$\mathsf{PA\text{-}UD}(1^\lambda, \mathcal{A})$ and $\mathsf{PA\text{-}UAD}(1^\lambda, \mathcal{A})$
$(\mathsf{mpk},\mathsf{msk},\mathsf{ek}) \leftarrow \mathsf{PAFE.Setup}(1^\lambda)$
$(\mathsf{mpk}^\star,\mathsf{ek}^\star) \leftarrow \mathcal{A}(1^\lambda)$
$(\mathsf{y}^\star,\pi^\star, i, j) \leftarrow \mathcal{A}^{\mathsf{PAFE.KeyGen}(\mathsf{mpk},\mathsf{msk},\,\cdot),\mathsf{PAFE.Enc}(\mathsf{mpk},\mathsf{ek},\cdot)}(1^\lambda, \mathsf{mpk})$
    ▷ $i$ corresponds to the $i$-th $\mathsf{PAFE.KeyGen}$ oracle query and
        $(f_i,\mathsf{sk}_{f,i},\mathsf{pk}_{f,i})$ is its associated information.
    ▷ $j$ corresponds to the $j$-th $\mathsf{PAFE.Enc}$ oracle query and
        $(\mathsf{x}_j,\mathsf{ct}_j)$ is its associated information.
If $(\mathsf{y}^\star \neq f_i(x_j) \wedge \mathsf{PAFE.VerifyDec}(\mathsf{mpk}, f_i,\ \mathsf{pk}_{f,i},\mathsf{ct}_j,\mathsf{y}^\star, \pi^\star) = 1)$
If $(\mathsf{y}^\star \neq f^\star(\mathsf{x}_j) \wedge \mathsf{PAFE.VerifyDec}(\mathsf{mpk}^\star, f^\star,\ \mathsf{pk}^\star_f,\mathsf{ct}_j,\mathsf{y}^\star,\pi^\star) = 1)$
        Output 1
Else output 0
```

Fig. 3. Public Auditability with Untrusted (Authority and) Decryptor

- *Functional key queries* QSKeyGen(f)*: $\mathcal{A}$ has unlimited and adaptive oracle access to the* $\mathsf{PAFE.KeyGen}(\mathsf{mpk},\mathsf{msk},f)$ *algorithm for input functions f of its choice. It receives the functional secret and public keys (sk_f ,pk_f).*
- *Functional public key queries* QPKeyGen(f)*: $\mathcal{A}$ has unlimited and adaptive oracle access to the* $\mathsf{PAFE.KeyGen}(\mathsf{mpk},\mathsf{msk},f)$ *algorithm for any input function f of its choice. It is given back a functional public key pk_f.*
- *Decryption queries* QDec(ct^b,f)*: $\mathcal{A}$ has unlimited adaptive access to an oracle for the* $\mathsf{PAFE.Dec}(\mathsf{mpk},f,\mathsf{sk}_f,\mathsf{ct})$ *algorithm, for any input ciphertext* ct *and any function f of its choice. If no Q(S/P)KeyGen query for f has been issued, $\mathcal{C}$ outputs $\perp$. Otherwise, $\mathcal{A}$ receives the function output $f(\mathsf{x}_b)$, where* ct_b=$\mathsf{PAFE.Enc}(\mathsf{mpk},\mathsf{ek},\mathsf{x}_b)$*, alongside a proof π about its correctness.*
- *Finalize: $\mathcal{A}$ provides its guess b' on the bit b and this procedure outputs the result β of the security game, according to the analysis given below.*

$\mathcal{A}$ wins in the game if $\beta = b$ and we remark that a naive adversary, by sampling randomly β has probability of winning equal to $\frac{1}{2}$. In case $\mathcal{A}$ has either issued (i) *QSKeyGen queries for some function f and also QEnc queries for a pair of values, $(\mathsf{x}_0,\mathsf{x}_1)$, such that $f(\mathsf{x}_0) \neq f(\mathsf{x}_1)$, or* (ii) *QDec query for ciphertext ct_b and function f and has issued a $\mathsf{ct}^b \leftarrow$ QEnc(x_0,x_1) query, where ct_0=PAFE.Enc(mpk,ek,x_0) and ct_1=PAFE.Enc(mpk,ek,x_1), such that $f(\mathsf{x}_0) \neq f(\mathsf{x}_1)$, then $\beta \leftarrow\!\!\$ \{0,1\}$. Otherwise $\beta = b'$. We denote the advantage that $\mathcal{A}$ has of winning as $Adv^{sec-PAFE}(\mathcal{A})$ and we say this* PAFE *is* secure *if for any PPT adversary $\mathcal{A}$, $Adv^{sec-PAFE}(\mathcal{A}) = |Pr[\beta = 1|b = 1] - Pr[\beta = 1|b = 0]| \leq negl$.*

As with FE, we can also have a selective version of the above game where the adversary declares all its encryption queries prior to other oracle accesses.

4.1 Public auditability definitions

There exist totally four different cases of public auditability, depending on which of the involved parties are adversarial and possibly colluding to trick the auditor

```
$\mathsf{PA\text{-}UED}(1^\lambda, \mathcal{A})$ and $\mathsf{PA\text{-}UEAD}(1^\lambda, \mathcal{A})$
$(\mathsf{mpk},\mathsf{msk},\mathsf{ek}) \leftarrow \mathsf{PAFE.Setup}(1^\lambda)$
$(\mathsf{mpk}^\star,\mathsf{ct}^\star,\{\mathsf{y}_i^\star,\pi_i^\star,f_i^\star,\mathsf{pk}_{f,i}^\star\}_{i\in[n]}) \leftarrow \mathcal{A}(1^\lambda)$
$(\mathsf{y}^\star,\pi^\star,\mathsf{ct}^\star,i) \leftarrow \mathcal{A}^{\mathsf{PAFE.KeyGen}(\mathsf{mpk},\mathsf{msk},\cdot)}(1^\lambda,\mathsf{mpk},\mathsf{ek})$
    $\triangleright$ $i$ corresponds to the $i$-th $\mathsf{PAFE.KeyGen}$ oracle query and
        $\{f_i,\mathsf{sk}_{f,i},\mathsf{pk}_{f,i}\}$ is its associated information.
If $(\nexists\, \mathsf{x} \in \mathcal{X}_\lambda : \mathsf{y}^\star = f_i(\mathsf{x}) \wedge \mathsf{PAFE.VerifyDec}(\mathsf{mpk}, f_i, \mathsf{pk}_{f,i}, \mathsf{ct}^\star, \mathsf{y}^\star){=}1)$
If $(\nexists\, \mathsf{x}' \in \mathcal{X}_\lambda : \forall i \in [n]\ \mathsf{y}_i^\star = f_i(\mathsf{x}') \wedge \mathsf{VerifyDec}(\mathsf{mpk}^\star,\mathsf{ct}^\star,f_i^\star, \mathsf{pk}_{f,i}^\star,\mathsf{y}_i^\star,\pi_i^\star){=}1)$
        Output 1
Else output 0
```

Fig. 4. Public Auditability with Untrusted Encryptor, (Authority, and) Decryptor.

into accepting an incorrect functional decryption result (see Table 1). In particular we consider the cases where: *(i)* only the decryptor is corrupted (PA-UD), *(ii)* both the encryptor and the decryptor are corrupted (PA-UED), *(iii)* both the key-generating authority and the decryptor are corrupted (PA-UAD), and *(iv)* the encryptor, the key-generating authority, and the decryptor are corrupted (PA-UEAD). Each corruption set results in a different definition of public auditability, defined below. At a high level, auditability ensures that, even though auditors have no decryption capabilities of their own, there is still a guarantee of (*at least* some level of) consistency between what they receive and what the decryptor computed. We note that this renders pointless to define this property for cases where the decryptor is honest (or the auditor itself is dishonest).

We provide game-based definition of public auditability for all four corruption cases. The two games for honest (resp. corrupted) authority are very similar, hence we depict them together in Figure 3 (resp. Figure 4). The lines shown in red correspond to the games modified for corrupted authorities, each time replacing the preceding line in black and we denote by the superscript $\star$ all adversary-provided elements.

Definition 9. [Public Auditability] *Let* $\mathsf{SET} \in \{\mathsf{UD},\mathsf{UAD},\mathsf{UED},\mathsf{UEAD}\}$ *and consider the experiments* PA-UD, PA-UAD *from Figure 3 and* PA-UED, PA-UEAD *from Figure 4. A PAFE for* $\mathcal{F}$ *is* SET-publicly auditable *against the corresponding set of corruptions if* $\forall \lambda \in \mathbb{N}$, $\forall \mathsf{x} \in \mathcal{X}_\lambda$, $\forall$*PPT adversaries* $\mathcal{A}$*, it holds that* $\Pr[\mathsf{PA\text{-}SET}(1^\lambda, \mathcal{A}) = 1] < negl(\lambda)$.

Based on the above definition we make the following observations. First, when the encryptor is honest, the auditor should be guaranteed to receive the exact function output corresponding to the function f_i of its choice and the actual plaintext x used by the honest encryptor. On the contrary, for cases PA-UED and PA-UEAD where the encryptor is not trusted, there is no guarantee that the ciphertexts were honestly computed. In these settings, the best the auditor can hope for is a notion of "consistency." I.e., there must exist some common plaintext element within the domain of f_i (or, in the case of multiple functions, the non-empty intersection of their respective domains) that evaluates to the output (or outputs) received and verified by the auditor.

Second, in the $\mathsf{PA\text{-}UD}$ experiment the condition $\mathsf{y}^\star \neq f_i(\mathsf{x}_j)$ could be equivalently written as $(\mathsf{y}^\star \neq \mathsf{PAFE.Dec}(\mathsf{mpk}, f_i, \mathsf{sk}_{f,i}, \mathsf{ct}_j))$. *However, this is not the case for PA-UAD.* In the former case the encryptor chooses an $\mathsf{x} \in \mathcal{X}_\lambda$ and encrypts honestly. In the latter case the encryptor may still intend to encrypt a valid plaintext x. However, the adversary (colluding authority and decryptor) may manipulate the generated keys so that even an honestly generated ciphertext is not decryptable into the image space of f_i.

5 PAFE Constructions from FE

Below we provide constructions for secure and publicly auditable PAFE schemes from secure FE schemes and other cryptographic primitives i.e., commitment schemes and NIWI proof systems. As in Section 4, we "group" our constructions together. However, now we do so based on whether the key-generating authority is trusted or not. Lines in blue in all figures of this section, depict *additional* steps/parts for when the encryptor is untrusted. Both auditable and regular correctness, for all our constructions, follow directly from the correctness of the underlying FE scheme, the correctness of the commitment scheme and the completeness of the employed NIWI proof systems. For each construction we provide proofs about its security and public auditability flavor. For readability reasons our proofs are delegated to the Appendix.

5.1 Auditability with trusted authority

PA-UD auditability. For this construction (Figure 5) we employ an FE scheme, a perfectly binding and computationally hiding commitment scheme, and a perfectly sound and computationally witness-indistinguishable NIWI proof system.

First, our functional public keys pk_f are computed as commitments of the corresponding sk_f. Second, $\mathsf{PAFE.ProveDec}$ produces a NIWI proof for the correctness of the decryption. The relation R_{UD} is shown in Figure 6. Symbol $\top$, included in this relation, is a fixed value from the supported domain of the commitment scheme Com, lying outside the domain of possible msk values that can be produced from $\mathsf{FE.Setup}$ (assuming, without loss of generality, that the domain of Com is a superset of the latter). This turns NIWI proofs into "OR"-proofs, allowing us to formally prove the IND-security of our scheme even in the presence of arbitrary oracle queries, without compromising auditability—since $\mathsf{PAFE.Setup}$, which is executed by the trusted authority, will never produce $\top$ as the msk. Below, we state Theorem 1 regarding the security and PA-UD public auditability of our construction, whose full proof we include in Appendix A.

Theorem 1. *Let* FE *be an IND-secure FE for a family of functions* $\mathcal{F}$. *Let* Com *be a perfectly binding and computationally hiding commitment scheme and* NIWI_d *a perfectly sound and computationally witness-indistinguishable NIWI for relation* R_{UD} *(Figure 6). Then, the PAFE scheme of Figure 5 for* $\mathcal{F}$ *is secure as per Definition 8 and UD-publicly auditable as per Definition 9.*

```
PAFE.Setup($1^\lambda$):
$s_1, s_2, u_d, u_e \leftarrow\$ \{0,1\}^\lambda$
pp $\leftarrow$ Com.Setup($1^\lambda$;$s_1$)
(msk,mpk,ek) $\leftarrow$ FE.Setup($1^\lambda$;$s_2$)
c$_d$ $\leftarrow$ Com.Commit(msk; $u_d$)
c$_e$ $\leftarrow$ Com.Commit(msk; $u_e$)
Return (pp,mpk,msk,ek,c$_d$,c$_e$)

PAFE.KeyGen(mpk,msk,$f$):
sk$_f$ $\leftarrow$FE.KeyGen(mpk,msk,$f$)
$r_f \leftarrow\$ \{0,1\}^\lambda$
pk$_f$ $\leftarrow$Com.Commit(sk$_f$; $r_f$)
Return (sk$_f$,$r_f$,pk$_f$)
```

```
PAFE.Enc(mpk,msk,ek,x,c$_e$,$u_e$):
ct $\leftarrow$ FE.Enc(ek,x;$s_e$), $s_e \leftarrow\$ \{0,1\}^\lambda$
$\pi_e$ $\leftarrow$NIWI$_e$.Prove(mpk,msk,ek,x,ct,c$_e$,$u_e$,$s_e$)
Return (ct,$\pi_e$)

PAFE.Dec(mpk,$f$,sk$_f$,ct):
y $\leftarrow$ FE.Dec(mpk,$f$,sk$_f$,ct)
Return y

$\pi_d$ $\leftarrow$NIWI$_d$.Prove(mpk,msk,$f$,sk$_f$,$r_f$,pk$_f$,ct,y,c$_d$,$u_d$)
Return $\pi_d$

PAFE.VerifyDec(mpk,$f$,pk$_f$,ct,y,c$_e$,c$_d$,$\pi_e$,$\pi_d$):
$b_1$ $\leftarrow$NIWI$_d$.Verify(mpk,$f$,pk$_f$,ct,y,c$_d$,$\pi_d$)
$b_2$ $\leftarrow$NIWI$_e$.Verify(mpk,ct,c$_e$,$\pi_e$)
Return $b_1 \wedge b_2$
```

Fig. 5. PAFE construction with untrusted (encryptor and) decryptor.

Relation $\mathsf{R}_{UED,d}$:	Relation $\mathsf{R}_{UED,e}$:
Statement: z_d = (mpk,f,pk$_f$,ct,y,c$_d$)	Statement: z_e = (mpk,ct,c$_e$)
Witness: w_d = (sk$_f$,r_f,$\top$,u_d)	Witness: w_e = (ek,x,s_e,$\top$,u_e)
$\mathsf{R}_{UED,d}(z_d, w_d) = 1$ iff:	$\mathsf{R}_{UED,e}(z_e, w_e) = 1$ iff:
(pk$_f$ $\leftarrow$ Com.Commit(sk$_f$; r_f)	ct=FE.Enc(mpk,ek,x; s_e)
$\wedge$ y $\leftarrow$ FE.Dec(mpk,f,sk$_f$,ct))	$\vee$c$_e$$\leftarrow$Com.Commit($\top$; u_e)
$\vee$ c$_d$ $\leftarrow$ Com.Commit($\top$; u_d)	

Fig. 6. Relations used in the constructions of Figure 5.

PA-UED auditability. To achieve public auditability against untrusted encryptor and decryptor we use the our PA-UD construction as a baseline, but additionally we require the encryptor to provide a NIWI proof for the validity of the ciphertext computation. The auditor now has to perform two NIWI verifications while executing the PAFE.VerifyDec algorithm. If both succeed, then it is convinced about the legitimacy of the received result. The additional elements this construction has with respect to the the PA-UD one are highlighted in blue in Figure 5 and the two relations $\mathsf{R}_{UED,e}$ and $\mathsf{R}_{UED,d}$ that need to be supported by the NIWI schemes are defined in Figure 6. Below we state Theorem 2 and we provide its proof in Appendix B.

Theorem 2. *Let* FE *be an IND-secure FE scheme for a family of functions $\mathcal{F}$. Let* Com *be a perfectly binding and computationally hiding commitment scheme and* NIWI$_d$,NIWI$_e$ *perfectly sound and computationally witness indistinguishable NIWI proof systems for relations $\mathsf{R}_{UED,d}$,$\mathsf{R}_{UED,e}$ (Figure 6). Then, the PAFE scheme of Figure 5 for $\mathcal{F}$ is adaptively secure under Definition 8 and publicly auditable against untrusted encryptors and decryptors as per Definition 9.*

5.2 Auditability with untrusted authority

Contrary to our previous constructions, since the authority can no longer be trusted to honestly run the setup and key-generation algorithms, our "OR" proof strategy no longer works. Instead, we adapt an approach from [10] based on replicating the FE computations, while accepting the majority of the functional decryptions as the correct output. In particular, our constructions use four instances of an IND-secure FE scheme, again combined with a perfectly binding and computationally hiding commitment scheme, and perfectly sound and computationally witness-indistinguishable NIWIs.

PA-UAD. To achieve public auditability in the presence of untrusted authority and decryptor, we augment the output of the $\mathsf{PAFE.KeyGen}$ algorithm to include a NIWI proof that guarantees at least three-out-of-four $\mathsf{FE.Setup}$ and $\mathsf{FE.KeyGen}$ instances have been generated honestly (Figure 7). Additionally, the output of the $\mathsf{PAFE.ProveDec}$ algorithm now states that at least two-out-of-the-four decryptions have been executed honestly. The corresponding relations $\mathsf{R}_{UAD,f}, \mathsf{R}_{UAD,d}$ that need to be supported by the NIWI schemes, are defined in Figure 8. What is more, $\mathsf{PAFE.Dec}$ now returns the majority of the functional decryptions—in case majority is not reached, $\mathsf{PAFE.Dec}$ returns $\perp$.

Auditability now is based on the fact that: (i) the encryptor is assumed to be trusted, meaning that it always encrypts the same message (regardless of how the encryption keys were originally generated), (ii) it is always guaranteed that *at least one* of the four FE instances is executed honestly with respect to key-generation, encryption, and decryption. To justify this in more detail, note that since three-out-of-four FE keys are computed correctly and two-out-of-four decryptions are performed correctly (and given the encryptor uses the same plaintext for all four ciphertexts), there is no way the sets of FE instances with correct keys and those with correct decryptions are disjoint.

More technically, this translates into the following: Suppose the decryptor provides the auditor with a fabricated $\mathsf{y}^\star$ that *does not* correspond to the output of the $\mathsf{FE.Dec}(\cdot)$ for any of the correctly computed $(\mathsf{mpk}, \mathsf{sk}_f)$. This would result in the decryptor breaking the correctness of the underlying FE scheme as all ciphertexts are also correctly computed. Hence, assuming the FE scheme is perfectly correct the only output y for which the auditor who runs $\mathsf{PAFE.VerifyDec}$ will accept is the correct functional output for the x encrypted by the encryptor. We now state the following theorem and we offload the formal proof to Appendix C.

Theorem 3. *Let* FE *be a secure FE scheme for a family of functions* $\mathcal{F}$*. Let* Com *be a perfectly binding and computationally hiding commitment scheme,* NIWI_f *and* NIWI_d *perfectly sound and computationally witness-indistinguishable NIWIs for* $\mathsf{R}_{UAD,d}$ *and* $\mathsf{R}_{UAD,d}$ *respectively (Figure 8). Then,* PAFE *in Figure 7 is adaptively secure under Definition 8 and publicly auditable against untrusted authority and decryptor as per Definition 9.*

PA-UEAD. Finally we focus on the "extreme" case where all protocol participants (authority, encryptor, and decryptor) collaborate to cheat the auditor

```
$\mathsf{PAFE.Setup}(1^\lambda)$:
For $i \in [4]$:
$(\mathsf{msk}_i,\mathsf{mpk}_i,\mathsf{ek}_i) \leftarrow \mathsf{FE.Setup}(1^\lambda;s_i)$
$s_i \leftarrow\$ \{0,1\}^\lambda$
$\mathsf{c}_f = \mathsf{Com.Commit}(1^{4 \cdot len}; u_f)$, $u_f \leftarrow\$ \{0,1\}^\lambda$
$\mathsf{c}_e = \mathsf{Com.Commit}(1^{len}; u_e)$ $u_e \leftarrow\$ \{0,1\}^\lambda$
$\mathsf{c}_d = \mathsf{Com.Commit}(1^{len}; u_d)$ $u_d \leftarrow\$ \{0,1\}^\lambda$
$\mathsf{msk} = \{\mathsf{msk}_i\}_{i\in[4]}$
$\mathsf{mpk} = (\{\mathsf{mpk}_i\}_{i\in[4]},\mathsf{c}_f,\mathsf{c}_e,\mathsf{c}_d)$
$\mathsf{ek} = \{\mathsf{ek}_i\}_{i\in[4]}$
Return $(\mathsf{mpk},\mathsf{msk},\mathsf{ek})$

$\mathsf{PAFE.KeyGen}(\mathsf{mpk},\mathsf{msk},f)$:
For $i \in [4]$:
$\mathsf{sk}_{f,i} \leftarrow \mathsf{FE.KeyGen}(\mathsf{mpk}_i,\mathsf{msk}_i,f; s_{k,i})$
$s_{k,i} \leftarrow\$ \{0,1\}^\lambda$
$\mathsf{pk}_{f,i} \leftarrow \mathsf{Com.Commit}(\mathsf{sk}_{f,i}; r_{f,i})$
$\mathsf{r}_{f,i} \leftarrow\$ \{0,1\}^\lambda$
$s_k = \{s_{k,i}\}_{i\in[4]}$
$r_f = \{r_{f,i}\}_{i\in[4]}$
$\mathsf{sk}_f = \{(\mathsf{sk}_{f,i}, r_{f,i}, s_{k,i})\}_{i\in[4]}$
$\mathsf{pk}_f = \{\mathsf{pk}_{f,i}\}_{i\in[4]}$
$\pi_f \leftarrow \mathsf{NIWI}_f.\mathsf{Prove}(\mathsf{mpk},\mathsf{msk},f,\mathsf{sk}_f,\mathsf{pk}_f,s_k,r_f)$
$\mathsf{pk}'_f = (\mathsf{pk}_f,\pi_f)$
Return $(\mathsf{sk}_f,\mathsf{pk}'_f)$
```

```
$\mathsf{PAFE.Enc}(\mathsf{mpk},\mathsf{ek},\mathsf{x})$:
For $i \in [4]$:
$\mathsf{ct}_i \leftarrow \mathsf{FE.Enc}(\mathsf{ek}_i,\mathsf{x};\mathsf{s}_{e,i})$
$\mathsf{s}_{e,i} \leftarrow\$ \{0,1\}^\lambda$
$s_e = \{s_{e,i}\}_{i\in[4]}$
$\pi_e \leftarrow \mathsf{NIWI}_e.\mathsf{Prove}(\mathsf{mpk},\mathsf{ek},\mathsf{x},\mathsf{ct},s_e)$
Return $\mathsf{ct} = (\{\mathsf{ct}_i\}_{i\in[4]},\pi_e)$

$\mathsf{PAFE.Dec}(\mathsf{mpk},f,\mathsf{sk}_f,\mathsf{ct})$:
For $i \in [4]$:
$\mathsf{y}_i \leftarrow \mathsf{FE.Dec}(\mathsf{mpk}_i,f,\mathsf{sk}_{f,i},\mathsf{ct}_i)$
If $\exists j_1 \neq j_2 \neq j_3 \in [4]$: $\mathsf{y}_{j_1} = \mathsf{y}_{j_2} = \mathsf{y}_{j_3}$
    Return $\mathsf{y}_1$
Return $\perp$

$\mathsf{PAFE.ProveDec}(\mathsf{mpk},f,\mathsf{sk}_f,\mathsf{pk}_f,\mathsf{ct},\mathsf{y})$:
$\pi_d \leftarrow \mathsf{NIWI}_d.\mathsf{Prove}(f,\mathsf{y},\mathsf{mpk},\mathsf{sk}_f,\mathsf{pk}_f,\mathsf{ct})$
Return $\pi_d$

$\mathsf{PAFE.VerifyDec}(\mathsf{mpk},\mathsf{pk}'_f,\mathsf{ct},\pi_d,\mathsf{y},f)$:
Parse $\mathsf{pk}'_f = (\mathsf{pk}_f, \pi_f)$
$b_1 \leftarrow \mathsf{NIWI}_f.\mathsf{Verify}(\mathsf{mpk}, \mathsf{pk}'_f, f)$
$b_2 \leftarrow \mathsf{NIWI}_e.\mathsf{Verify}(\mathsf{mpk},\mathsf{ct})$
$b_3 \leftarrow \mathsf{NIWI}_d.\mathsf{Verify}(\mathsf{mpk},\mathsf{pk}_f,\mathsf{ct},\mathsf{y},\pi_d,f)$
Return $b_1 \wedge b_2 \wedge b_3$
```

Fig. 7. PAFE construction with untrusted encryptor, authority, and decryptor.

into accepting an incorrect functional output. To achieve public auditability in this case, we use the PA-UAD construction as a baseline but we augment the output of the encryption to include a NIWI proof as well, following a similar approach as the one we adopted to go from PA-UD to PA-UED. However, given the multiple FE instances we need to further modify our technique (Figure 7).

Overall, we use NIWIs to prove that at least three-out-of-the-four executions of the FE.Setup and FE.KeyGen algorithms have been executed honestly. Likewise, for at least two-out-of-the-four executions of the FE.Enc and at least three-out-of-the four executions ofFE.Dec algorithms. The above are depicted in relations $R_{UEAD,f}$, $R_{UEAD,e}$ and $R_{UEAD,d}$ in Figure 8. Unlike in the PA-UAD case, here we additionally consider the encryptor to be untrusted therefore we cannot use the same reasoning to guarantee that at least for one FE instance all the algorithms (FE.Setup,FE.KeyGen,FE.Enc,FE.Dec), have been correctly executed. Instead, we reason about the properties of our construction as follows:

First, $R_{UEAD,f}$ and $R_{UEAD,d}$ both require that pk_f is computed honestly, for their respective indices, which "ties them together". On the other hand, the encryption algorithm is not tied to the pk_f. However, having a three-out-of-four threshold for the NIWI proof regarding functions key generation suffices

Relation $\mathsf{R}_{UAD,f}$:

Statement: z_f=($\{\mathsf{mpk}_i,f_i,\mathsf{pk}_{f,i},\mathsf{ct}_i\}_{i\in[4]}$); Witness: w_f=($\{\mathsf{msk}_i,\mathsf{sk}_{f,i},r_{f,i},r_{k_i},s_i\}_{i\in[4]}$)

$\mathsf{R}_{UAD,f}(z_f,w_f)$=1 iff either of the following conditions hold:

(1) - $\forall j \in [4]$: $\mathsf{pk}_{f,j} \leftarrow \mathsf{Com}(\mathsf{sk}_{f,j};r_{f,j}) \wedge (\mathsf{mpk}_j,\mathsf{msk}_j) \leftarrow \mathsf{FE.Setup}(1^\lambda;s_j)$
$\wedge\ \mathsf{sk}_{f,j} \leftarrow \mathsf{FE.KeyGen}(\mathsf{mpk}_j,\mathsf{msk}_j,f_j;r_{k_j})$

(2) - $\exists A_1 \subset [4]$, with $|A_1|=3$, s.t. $\forall j \in A_1$: $\mathsf{pk}_{f,j} \leftarrow \mathsf{Com.Commit}(\mathsf{sk}_{f,j};r_{f,j})$
$\wedge\ (\mathsf{mpk}_j,\mathsf{msk}_j) \leftarrow \mathsf{FE.Setup}(1^\lambda;s_j) \wedge \mathsf{sk}_{f,j} \leftarrow \mathsf{FE.KeyGen}(\mathsf{mpk}_j,\mathsf{msk}_j,f;r_{k_j})$

Relation $\mathsf{R}_{UAD,d}$:

Statement: z_d=($\{\mathsf{mpk}_i,f_i,\mathsf{pk}_{f,i},\mathsf{ct}_i\}_{i\in[4]}$,$\mathsf{y}$,$\pi_d$) Witness: w_d=($\{\mathsf{sk}_{f,i},r_{f,i}\}_{i\in[4]}$)

$\mathsf{R}_{UAD,d}(z_d,w_d)$=1 iff either of the following conditions hold:

(1) $\forall k \in [4]$: $\mathsf{pk}_{f_k} \leftarrow \mathsf{Com}(\mathsf{sk}_{f_k};r_{f_k}) \wedge \mathsf{y} \leftarrow \mathsf{FE.Dec}(\mathsf{mpk}_k,f,\mathsf{sk}_{f_k},\mathsf{ct}_k)$

(2) $\exists A_2 \subset [4]$, with $|A_2|=2$, s.t. $\forall k \in A_2$: $\mathsf{pk}_{f_k} \leftarrow \mathsf{Com}(\mathsf{sk}_{f_k};r_{f_k})$
$\wedge\ \mathsf{y} \leftarrow \mathsf{FE.Dec}(\mathsf{mpk}_k,f,\mathsf{sk}_{f_k},\mathsf{ct}_k) \wedge \mathsf{c}_f \leftarrow \mathsf{Com}(\{\mathsf{sk}_{f,i}\}_{i\in[4]};u_f) \wedge \mathsf{c}_d \leftarrow \mathsf{Com}(0^{len};u_d)$

Relation $\mathsf{R}_{UEAD,f}$:

Statement: z_f=$\{\mathsf{mpk}_i,f_i,\mathsf{pk}_{f,i},\mathsf{ct}_i\}_{i\in[4]}$,$\mathsf{c}_d,\mathsf{c}_f$);

Witness:w_f=($\{\mathsf{msk}_i,\mathsf{sk}_{f,i},r_{f,i},s_i\}_{i\in[4]},\mathsf{y},u_d,u_f$)

$\mathsf{R}_{UEAD,f}(z_f,w_f)$=1 iff either of the following conditions hold:

(1) - ($\forall j \in [4]$: $\mathsf{pk}_{f,j} \leftarrow \mathsf{Com.Commit}(\mathsf{sk}_{f,j};r_{f,j}) \wedge (\mathsf{mpk}_j,\mathsf{msk}_j) \leftarrow \mathsf{FE.Setup}(1^\lambda;s_j)$
$\wedge\ \mathsf{sk}_{f,j} \leftarrow \mathsf{FE.KeyGen}(\mathsf{mpk}_j,\mathsf{msk}_j,f_j;r_{f,j})$) $\wedge\ \mathsf{c}_f \leftarrow \mathsf{Com.Commit}(1^{4\cdot len};u_f)$

(2) - ($\exists A_1 \subset [4]$, with $|A_1|=3$, s.t. $\forall j \in A_1$:
$\mathsf{pk}_{f,j} \leftarrow \mathsf{Com.Commit}(\mathsf{sk}_{f,j};r_{f,j}) \wedge (\mathsf{mpk}_j,\mathsf{msk}_j) \leftarrow \mathsf{FE.Setup}(1^\lambda;s_j)$
$\wedge\ \mathsf{sk}_{f,j} \leftarrow \mathsf{FE.KeyGen}(\mathsf{mpk}_j,\mathsf{msk}_j,f;r_{f,j}))\wedge \mathsf{c}_f \leftarrow \mathsf{Com.Commit}(0^{4\cdot\lambda};u_f)$
$\wedge\ \exists\ \mathsf{y} \in \mathcal{X}_\lambda$ such that $\forall\ i \in [4]$: $\mathsf{y} \leftarrow \mathsf{FE.Dec}(\mathsf{mpk}_i,f_i,\mathsf{sk}_{f,i},\mathsf{ct}_i)$

Relation $\mathsf{R}_{UEAD,e}$:

Statement: z_e=($\{\mathsf{mpk}_i, \mathsf{ct}_i\}_{i\in[4]}$,$\mathsf{y}$,$\mathsf{c}_e$); Witness: w_e=($\{\mathsf{ek}_i,\mathsf{x}_i,s_{e,i}\}_{i\in[4]},u_e$)

$\mathsf{R}_{UEAD,e}(z_e,w_e)$=1 iff either of the following conditions hold:

(1) $\forall k \in [4]$:
$\mathsf{ct}_j = \mathsf{FE.Enc}(\{\mathsf{ek}_j,\mathsf{mpk}_j\},\mathsf{x};s_{e,j}) \wedge \mathsf{c}_f \leftarrow \mathsf{Com.Commit}(1^{4\cdot len};u_f)$
$\wedge\ \mathsf{c}_e \leftarrow \mathsf{Com.Commit}(1^{len};u_e)$

(2) $\exists A_2 \subset [4]$, with $|A_2|=2$, s.t. $\forall k \in A_2$:
$\mathsf{ct}_k = \mathsf{FE.Enc}(\{\mathsf{ek}_k,\mathsf{mpk}_k\},\mathsf{x};s_{e,k}) \wedge \mathsf{c}_e \leftarrow \mathsf{Com.Commit}(0^{len};u_e)$

Relation $\mathsf{R}_{UEAD,d}$:

Statement: z_d=($\{\mathsf{mpk}_i,f_i,\mathsf{pk}_{f,i},\mathsf{ct}_i\}_{i\in[4]}$,$\mathsf{y}$,$\pi_d$); Witness: w_d=($\{\mathsf{sk}_{f,i},r_{f,i}\}_{i\in[4]}$)

$\mathsf{R}_{UEAD,d}(z_d,w_d)$=1 iff either of the following conditions hold:

(1) $\forall k \in [4]$:
$\mathsf{pk}_{f_k} \leftarrow \mathsf{Com.Commit}(\mathsf{sk}_{f_k};r_{f_k}) \wedge \mathsf{y} \leftarrow \mathsf{FE.Dec}(\mathsf{mpk}_k,f_k,\mathsf{sk}_{f_k},\mathsf{ct}_k)$
$\wedge\ \mathsf{c}_e \leftarrow \mathsf{Com.Commit}(1^{len};u_f)$

(2) $\exists A_3 \subset [4]$, with $|A_3|=3$, s.t. $\forall k \in A_3$:
$\mathsf{pk}_{f_k} \leftarrow \mathsf{Com.Commit}(\mathsf{sk}_{f_k};r_{f_k}) \wedge \mathsf{y} \leftarrow \mathsf{FE.Dec}(\mathsf{mpk}_k,f_k,\mathsf{sk}_{f_k},\mathsf{ct}_k)$
$\wedge\ \mathsf{c}_d \leftarrow \mathsf{Com.Commit}(0^{len};u_d)$

Fig. 8. Relations used in the constructions for secure PA-UAD and PA-UEAD PAFE.

to guarantee majority in $\mathsf{PAFE.Dec}$. Finally, note that $R_{UEAD,f}$ enforces that all four decryptions return the same result which guarantees that the auditor receives a valid function output. Recall that, since the encryptor is not assumed honest in this setting, we cannot guarantee that the output y is the correct functional output for *the* plaintext x used by for all four ciphertexts; just a "consistent" explanation for all four of them (see Section 4.1).

One downside of our final construction is that it only satisfies the selective version of Definition 8, where the adversary issues all encryption queries before any other oracle access (but after receiving the output of $\mathsf{PAFE.Setup}$). To see why this is necessary, observe that condition (2) of relation $R_{UEAD,f}$ requires knowledge of all ciphertexts ct_i for the NIWI proof generation. Similarly to before, we state the following theorem and delegate the proof for our PA-UEAD scheme to Appendix D.

Theorem 4. *Let FE be an IND-secure FE scheme for a family of functions $\mathcal{F}$. Let Com be a perfectly binding and computationally hiding commitment scheme, NIWI_f, NIWI_e, NIWI_d computationally witness-indistinguishable and perfectly sound NIWIs for relations $R_{UEAD,e}$, $R_{UEAD,f}$, and $R_{UEAD,d}$ shown in Figure 8. Then, PAFE in Figure 7 is selectively secure under Definition 8 and publicly auditable against untrusted encryptor, authority, and decryptor as per Definition 9.*

6 Relations Among PA definitions

Here, we investigate any implications between the public auditability definitions. The strongest out of the four is the one where all entities are untrusted, whereas the weakest one is the one where just the decryptor is untrusted. A natural assumption would be that the strongest implies any weaker definition. However, this is not the case and below we elaborate more on the reason why.

PA-UAD $\Longrightarrow$ PA-UD: In order to prove this we consider the contrapositive, that is assuming the existence of an adversary $\mathcal{A}$ that wins in the PA-UD with non-negligible probability, we will construct an adversary $\mathcal{A}'$ that wins in the PA-UAD with also non-negligible probability. $\mathcal{A}$, additionally to the needs of $\mathcal{A}'$ requires an honestly generated mpk and oracle access to the key-generating algorithm. $\mathcal{A}'$ runs $\mathsf{PAFE.Setup}$ honestly once and provides mpk to $\mathcal{A}$, and whenever the latter issues a QKeyGen query $\mathcal{A}'$ simulates the random oracle. It runs $\mathsf{PAFE.KeyGen}$ honestly, forwards the output to $\mathcal{A}$, and stores the input-output information so it can answer future repeated queries. Clearly both $\mathcal{A}$ and $\mathcal{A}'$ have equal advantage of winning their respective games, which concludes our analysis.

PA-UEAD $\Longrightarrow$ PA-UED: Similarly, to prove this we consider the contrapositive, that is assuming the existence of an adversary $\mathcal{A}$ who wins in the PA-UED with non-negligible probability, we construct an adversary $\mathcal{A}''$ that wins in the PA-UEAD with also non-negligible probability. The proof is similar to the previous reduction as $\mathcal{A}''$ operates exactly as $\mathcal{A}'$. Therefore, both $\mathcal{A}$ and $\mathcal{A}''$ have equal advantage of winning their respective games.

For the remaining relations, namely **PA-UED** $\implies$ **PA-UD** and **PA-UEAD** $\implies$ **PA-UAD** we observe that the winning conditions for adversaries on either side of the implications are different. Specifically, when the encryptor is trusted it produces a legitimate ciphertext that is decryptable to a functional output. Thus, public auditability is satisfied only if the auditor receives that exact function output. However, when the encryptor is untrusted, it suffices for the ciphertext to be decryptable to any function output. This discrepancy in the winning conditions obscures the remaining relations significantly.

7 Publicly Auditable Inner-Product MIFE

Here we present a publicly auditable MIFE scheme for inner-product functionalities, i.e., for weighted sums between plaintext vectors $\mathsf{x}{=}\{\mathsf{x}_i\}_{i\in[n]} \in \mathbb{Z}_q^n$ and weights $\{w_i\}_{i\in[n]} \in \mathbb{Z}_q^n$. We note here that the main difference between FE and MIFE is that the latter allows computing a function over multiple (n) ciphertexts, all of which are individually computed with different encryption keys.

To realise our construction, we employ the existing inner-product multi-input scheme of [7]. At a high level, that construction produces "ElGamal-style" ciphertexts, as follows. Given a vector of inputs $\mathsf{x}{=}\{\mathsf{x}_i\}_{i\in[n]} \in \mathbb{Z}_q^n$, the corresponding ciphertexts are of the form $(g^r, h^r, \{g^{\mathsf{x}_i} \cdot h_i^r\}_{i\in[n]})$, for encryption key $\mathsf{ek}{=}(\{\mathsf{s}_i,\mathsf{t}_i\}_{i\in[n]})$, and master public key $\mathsf{mpk}{=}\ (\mathbb{G}, g, h, \{h_i\}_{i\in[n]})$. The decryptor multiplies all ciphertexts and divides the product by $(g^r)^{\mathsf{sk}_{f,g}} \cdot (h^r)^{\mathsf{sk}_{f,h}}$, where $\mathsf{sk}_{f,g} = \Sigma_{i=1}^n s_i \cdot w_i$ and $\mathsf{sk}_{f,h} = \Sigma_{i=1}^n t_i \cdot w_i$, and then solves the discrete logarithm problem to acquire the inner product output (assuming a "small" domain for x_i and n, so that the final discrete logarithm can be computed efficiently [7]).

We now present our PA MIFE scheme that achieves public auditability in the presence of untrusted decryptors (PA-UD), using a non-interactive zero-knowledge argument (NIZK) [15] to produce a proof of correct decryption. Referring to the construction of Figure 9, we observe that the decryptor needs only give the term $(g^r)^{\mathsf{sk}_{f,g}} \cdot (h^r)^{sk_{f,h}}$ to potential auditors and convince them about the fact that sk_f has been used appropriately to generate this term. This can be done by proving discrete logarithm relations between what the verifier already knows and the information received from the decryptor, i.e., proving knowledge of a common discrete logarithm between multiple DDH tuples.

This can be done via a general-purpose NIZK or even a Σ-protocol for DDH tuples (e.g., see [31, Ch. 5.2]). In the second case, the resulting construction would be very efficient and would not require trusted CRS generation. To satisfy the security definition, the NIZK should be fully zero-knowledge; if instantiated with a $\Sigma-$protocol we assume the non-interactive version based on the Fiat-Shamir heuristic is used which also makes it zero-knowledge against arbitrary verifiers. Formally, the relation for the NIZK , in Camenisch-Stadler notation [17], is:

$$\mathbf{PK}\{(x,r)\mathbf{:}(g,h,g^r,h^r,g^{\Sigma_{i=1}^n s_i\cdot w_i},h^{\Sigma_{i=1}^n t_i\cdot w_i},g^{r\cdot\Sigma_{i=1}^n s_i\cdot w_i},h^{r\cdot\Sigma_{i=1}^n t_i\cdot w_i},g^{u_d}\cdot h^{r_d})\}$$

We state the following theorem and delegate the proof for our publicly auditable MIFE scheme to Appendix E.

```
$\mathsf{PAFE.Setup}(1^\lambda, 1^n)$:
    Choose cyclic group $\mathbb{G}$
    of prime order $q > 2^\lambda$
    with generators $g, h \leftarrow\$ \mathbb{G}$
    For $i \in [n]$:
    $s_i, t_i \leftarrow\$ \mathbb{Z}_p$
    $h_i = g^{s_i} \cdot h^{t_i}$
    $\mathsf{ek} = \{(s_i, t_i)\}_{i\in[n]}$
    $\mathsf{msk} = \{(s_i, t_i)\}_{i\in[n]}$
    $c_d = g^{\Sigma_{i=0}^n s_i \cdot t_i} \cdot h^{\Sigma_{i=0}^n s_i \cdot t_i}$
    $\mathsf{mpk} = (\mathbb{G}, g, h, c_d, \{h_i\}_{i\in[n]})$

$\mathsf{PAFE.KeyGen}(\mathsf{msk}', \mathsf{mpk}', \{w_i\}_{i\in[n]})$:
    Parse $\mathsf{msk}' = \{(s_i, t_i)\}_{i\in[n]}$
    Parse $\mathsf{mpk}' = (\mathbb{G}, g, h, c_d, \{h_i\}_{i\in[n]})$
    $sk_f = (\Sigma_{i=1}^n s_i \cdot w_i, \Sigma_{i=1}^n t_i \cdot w_i)$
    $pk_f = (g^{\Sigma_{i=1}^n s_i \cdot w_i}, h^{\Sigma_{i=1}^n t_i \cdot w_i})$

$\mathsf{PAFE.Enc}(\mathsf{mpk}', \mathsf{ek}', \{\mathsf{x}_i\}_{i\in[n]})$:
    Parse $\mathsf{ek}' := \{s_i, t_i\}_{i\in[n]}$
    Parse $\mathsf{mpk}' = (\mathbb{G}, g, h, c_d, \{h_i\}_{i\in[n]})$
    $r_e \leftarrow\$ \mathbb{Z}_p$
    $\mathsf{ct} = (g^r, h^r, \{g^{\mathsf{x}_i} \cdot h_i^r\}_{i\in[n]})$
```

```
$\mathsf{PAFE.Dec}(\mathsf{mpk}', \{w_i\}_{i\in[n]}, \mathsf{sk}'_f, \mathsf{ct}')$:
    Parse $\mathsf{mpk}' = (\mathbb{G}, g, h, c_d, \{h_i\}_{i\in[n]})$
    Parse $\mathsf{sk}'_f = (\Sigma_{i=1}^n s_i \cdot w_i, \Sigma_{i=1}^n t_i \cdot w_i)$
    Parse $\mathsf{ct}' = (g^r, h^r, \{g^{\mathsf{x}_i} \cdot h_i^r\}_{i\in[n]})$
    $\mathsf{y} = \frac{\prod_{i=1}^n g^{x_i} \cdot h_i^r}{g^{r\cdot(\Sigma_{i=1}^n s_i \cdot w_i)} \cdot h^{r\cdot(\Sigma_{i=1}^n t_i \cdot w_i)}}$

$\mathsf{PAFE.ProveDec}(\mathsf{mpk}', \{w_i\}_{i\in[n]}, \mathsf{sk}'_f, \mathsf{pk}'_f, \mathsf{ct}', \mathsf{y}')$:
    Parse $\mathsf{mpk}' = (\mathbb{G}, g, h, c_d, \{h_i\}_{i\in[n]})$
    Parse $\mathsf{sk}'_f = (\Sigma_{i=1}^n s_i \cdot w_i, \Sigma_{i=1}^n t_i \cdot w_i)$
    Parse $\mathsf{pk}_f{}' = (g^{\Sigma_{i=1}^n s_i \cdot w_i}, h^{\Sigma_{i=1}^n t_i \cdot w_i})$
    Parse $\mathsf{ct}' = (g^r, h^r, \{g^{\mathsf{x}_i} \cdot h_i^r\}_{i\in[n]})$
    $\pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{mpk}', \{w_i\}_{i\in[n]}, \mathsf{sk}'_f, \mathsf{pk}'_f, \mathsf{ct}', \mathsf{y}')$
    Return $\pi$

$\mathsf{PAFE.VerifyDec}(\mathsf{mpk}', \{w_i\}_{i\in[n]}, \mathsf{pk}'_f, \mathsf{ct}', \mathsf{y}', \pi')$:
    Parse $\mathsf{mpk}' = (\mathbb{G}, g, h, c_d, \{h_i\}_{i\in[n]})$
    Parse $\mathsf{pk}_f{}' = (g^{\Sigma_{i=1}^n s_i \cdot w_i}, h^{\Sigma_{i=1}^n t_i \cdot w_i})$
    Parse $\mathsf{ct}' = (g^r, h^r, \{g^{\mathsf{x}_i} \cdot h_i^r\}_{i\in[n]})$
    Return $\mathsf{NIZK.Verify}(\mathsf{mpk}', \{w_i\}_{i\in[n]}, \mathsf{pk}'_f, \mathsf{ct}', \mathsf{y}', \pi')$
```

Fig. 9. Publicly auditable MIFE for inner products with untrusted decryptor.

Theorem 5. *The construction depicted in Figure 9 is a PAFE scheme for inner-product functionalities. It is secure as per Definition 8 and is publicly auditable as per Definition 9, assuming that the MIFE scheme of [7] is IND-secure as per Definition 6 and the employed NIZK is computationally sound.*

8 Conclusion & Discussion

In this work we introduced public auditability in the context of functional encryption. We defined four flavors of PA, for each different corresponding corruption set among the participating parties, and presented corresponding constructions that achieve these definitions, as well as a novel indistinguishability security definition for PAFE. Our constructions rely on secure FE, commitments, and NIWI schemes to force the parties to prove their correct behavior. Finally, we proposed a multi-input PAFE scheme that supports linear combination functionalities expressed as vector inner-products. It builds upon previous MIFE schemes that produce "El Gamal"-style ciphertexts that are amenable to Σ-protocols, thus being potentially very efficient for use in practice.

Our work leaves many possible directions for future research in this topic. First, it would be of interest to design an adaptively secure PA-UEAD PAFE scheme, since our presented construction for this setting is only selectively secure. In this aspect, we believe it is possible to consider a different type of model relaxation of security, i.e., with "static" function key queries. This refers to an adversary who specifies mutually exclusive sets of functions for which it either requests functional secret keys or solely public ones. This setting is incomparable to selective security but it seems more applicable to several real-world applications like the ones we described in the introduction of the paper. Another direction would be to design and implement practical PAFE schemes in any of the auditability settings, at least for more expressive functionalities (e.g., quadratic functions from [11, 24]). Finally, it is worth exploring other use cases and applications for more efficient PAFE schemes, e.g., in the context of auditable cryptocurrencies [18, 26, 29] and auditable blockchain storage [32].

Acknowledgements

We would like to thank the anonymous reviewers for their constructive feedback. This work was partially supported by Hong Kong RGC under grant 16200721.

Appendix

A Proof of Theorem 1

We prove the PAFE security game indistinguishable regardless of the challenger bit. We define multiple Hybrids to go from the execution of the PAFE security game with $b = 0$ to the execution with $b = 1$ and prove them subsequently indistinguishable. We state the advantage that the adversary has during each transformation and provide the total advantage at the end of our analysis.

Note that we exclude from our analysis adversarial strategies that trivially win the PAFE security game (by violating its winning conditions). This means that if the adversary issues a series of queries like $(^\star)$ or $(^{\star\star})$ the advantage of the adversary is reduced to 0, from the PAFE security game (Def. 8).

$$\begin{array}{r|c} (^\star) : \text{QEnc}(\mathsf{x}_0,\mathsf{x}_1),\ \text{QSKeyGen}(f) & \text{such that} \\ (^{\star\star}) : \text{QEnc}(\mathsf{x}_0,\mathsf{x}_1)\rightarrow\mathsf{ct},\ \text{QPKeyGen}(f),\ \text{QDec}(\mathsf{ct},f) & f(\mathsf{x}_0)\neq f(\mathsf{x}_1) \end{array}$$

Now, observe that we can divide all remaining possible, non-trivially-winning, strategies into two cases, based on whether the adversary issues QDec$(\cdot,\cdot)$ queries (case *(i)*) or not (case *(ii)*).

Intuitively by making such a division first we "exploit" the fact that adversaries who do not issue QDec$(\cdot,\cdot)$ queries (case *(ii)*), essentially degenerate into FE-type adversaries. The only exception is that they can also have access to functional public keys (which are computationally hiding commitments). On the other hand, we know that the adversary in case *(i)* will issue at least one non-trivially-violating QDec$(\mathsf{ct},\cdot)$ query, for QEnc$(\mathsf{x}_0,\mathsf{x}_1)\rightarrow\mathsf{ct}$. This allows us to define hybrids over the total number of QEnc queries that are subsequently different in just a single output of the QEnc(x_0,x_1)$\rightarrow\mathsf{ct}^b$ query (based on the challenger bit) and prove them indistinguishable. In more detail, we present our analysis for the two cases below:

*Proof (**Security**).*

Case (i): We assume $\mathcal{A}_{\mathsf{PAFE}}$ issues at least one QDec($\cdot,\cdot$) query. We prove indistinguishability of the game that $\mathcal{A}_{PAFE}$ plays when $b=0$ and $b=1$ through a series of hybrids. Below we define the hybrids and prove them consecutively indistinguishable. The challenger bit is represented in the game/hybrid exponents.

Hybrid $\mathcal{G}^0_{UD}$: It is the security game when $b=0$.

Hybrid $\mathcal{H}^0_{UD,1}$: It is exactly the same game as $\mathcal{G}^0_{UD}$ except for the computation of the c_d. In $\mathcal{G}^0_{UD}$ $\mathsf{c}_d \leftarrow \mathsf{Com.Commit}(\mathrm{msk},; u_d)$, whereas in $\mathcal{H}^0_{UD,1}$ $\mathsf{c}'_d \leftarrow \mathsf{Com.Commit}(\top; u_d)$. From the hiding property of the employed commitment scheme no PPT adversary who sees a commitment can identify the committed value. Thus, $\mathcal{G}^0_{UD} \approx \mathcal{H}^0_{UD,1}$ and more specifically, $Adv^{\text{Distinguish}}_{\mathcal{G}^0_{UD}-\mathcal{H}^0_{UD,1}}(\mathcal{A}_{\mathsf{PAFE}}) = Adv^{\text{Com-Hidding}}(\mathcal{A}_{\mathsf{PAFE}})$.

Hybrid $\mathcal{H}^0_{UD,2}$: It is exactly the same game as $\mathcal{H}^0_{UD,1}$ except for the computation of π_d. In $\mathcal{H}^0_{UD,1}$ $\pi_d \leftarrow \mathsf{NIWI}_d.\mathsf{Prove}(\mathsf{mpk},\mathsf{msk},f,\mathsf{sk}_f,r_f,\mathsf{pk}_f,\mathsf{ct},\mathsf{y},\mathsf{c}_d,u_d)$ using the first condition for relation $\mathsf{R}_{UD,d}$, whereas in $\mathcal{H}^0_{UD,2}$, using the second condition of $\mathsf{R}_{UD,d}$, $\pi'_d \leftarrow \mathsf{NIWI}_d.\mathsf{Prove}(\mathsf{mpk},\perp,f,\perp,\perp,\mathsf{pk}_f,\mathsf{ct},\mathsf{y},\mathsf{c}_d,u_d)$ respectively. From the witness indistinguishability property of NIWI$_d$ no PPT adversary can distinguish between which condition is satisfied for the generation of π_d. Thus, $\mathcal{H}^0_{UD,1} \approx \mathcal{H}^0_{UD,2}$ and more specifically, $Adv^{\text{Distinguish}}_{\mathcal{H}^0_{UD,1}-\mathcal{H}^0_{UD,2}}(\mathcal{A}_{\mathsf{PAFE}}) = Adv^{WI}_{NIWI}(\mathcal{A}_{\mathsf{PAFE}})$.

Hybrid $\mathcal{H}^0_{UD,3}$: It is exactly the same game as $\mathcal{H}^0_{UD,2}$ except for the computation of the y. In this case, we change y to be $\mathsf{y} = f(\mathsf{x})$ instead of $\mathsf{y} \leftarrow \mathsf{FE.Dec}(\mathsf{mpk},f,\mathsf{sk}_f,\mathsf{ct})$. Remember that for $\mathcal{A}_{\mathsf{PAFE}}$ to have non-negligible chance of winning in its game, it must be that for all functions f that $\mathcal{A}_{\mathsf{PAFE}}$ issues a QSKeyGen(f) query, for all $\mathsf{ct} \leftarrow$ QEnc(x_0,x_1): $f(\mathsf{x}) = f(\mathsf{x}_0) = f(\mathsf{x}_1)$. Additionally and similarly, for all functions f for which $\mathcal{A}_{\mathsf{PAFE}}$ has issued QPKeyGen(f) and QDec(ct,f) queries, where $\mathsf{ct}\leftarrow$ QEnc(x_0,x_1), it must be that $f(\mathsf{x}) = f(\mathsf{x}_0) = f(\mathsf{x}_1)$. In any other case by the restrictions of the security game for PAFE $Adv^{sec-PAFE}\big(\mathcal{A}_{\mathsf{PAFE}}(1^\lambda)\big) = 0$. Since $\mathcal{A}_{\mathsf{PAFE}}$ cannot win in any of these two games with non-negligible advantage unless $f(\mathsf{x}_0) = f(\mathsf{x}_1)$, $\mathcal{H}^0_{UD,2} \approx \mathcal{H}^0_{UD,3}$ and more specifically, $Adv^{\text{Distinguish}}_{\mathcal{H}^0_{UD,2}-\mathcal{H}^0_{UD,3}}(\mathcal{A}_{\mathsf{PAFE}}) = 0$.

Hybrid $\mathcal{H}^0_{UD,4}$: In this game we make the following change: the challenger samples $j' \leftarrow\$ \{0,\cdots,m+1\}$, initializes a counter $j=0$, and when $\mathcal{A}_{\mathsf{PAFE}}$ issues an encryption query, the challenger sets $j=j+1$ and returns ct^b_j (we denote that query as QEnc($\mathsf{x}_{0,j}$,$\mathsf{x}_{1,j}$), more concretely). Now, when $\mathcal{A}_{\mathsf{PAFE}}$ issues a QPKeyGen(f) query, $\mathcal{C}$ checks whether $f(\mathsf{x}_{0,j'}) \neq f(\mathsf{x}_{1,j'})$. If so, it samples z_f, $r_f \leftarrow\$ \mathbb{Z}_p$ and computes $\mathsf{pk}_f \leftarrow \mathsf{Com.Commit}(\mathsf{z}_f,r_f)$. Remember that since $f(\mathsf{x}_{0,j'}) \neq f(\mathsf{x}_{1,j'})$ the adversary cannot issue a QSKeyGen(f) or a QDec($\mathsf{ct}^b_{j'}$,f) query — for that particular ciphertext. In such cases $\mathcal{A}_{\mathsf{PAFE}}$ would trivially diminish its advantage to 0, contradicting our assumption that it has non-negligible advantage ϵ in winning the security game for PAFE. Therefore, from the hiding property of the underlying commitment scheme, similarly to $\mathcal{G}^0_{UD} \approx \mathcal{H}_{UD,1}$, we get that $\mathcal{H}^0_{UD,3} \approx \mathcal{H}^0_{UD,4}$ and more specifically, $Adv^{\text{Distinguish}}_{\mathcal{H}^0_{UD,3}-\mathcal{H}^0_{UD,4}}(\mathcal{A}_{\mathsf{PAFE}}) = Adv^{\text{Com-Hidding}}(\mathcal{A}_{\mathsf{PAFE}})$.

Hybrid $\mathcal{H}^b_{UD,5.j}$: We now define a series of hybrids, indexed by j. In these hybrids we make the following change: the challenger samples $b \leftarrow\!\!\$\ \{0,1\}$ and when $\mathcal{A}_{\mathsf{PAFE}}$ issues a QEnc(x_0,x_1) query $\mathcal{C}$ returns $\mathsf{ct}^0 \leftarrow \mathsf{PAFE.Enc(mpk,ek,x_0)}$, if $j < j'$, $\mathsf{ct}^1 \leftarrow \mathsf{PAFE.Enc(mpk,ek,x_1)}$, if $j > j'$, and $\mathsf{ct}^b \leftarrow \mathsf{PAFE.Enc(mpk,ek,x}_b)$, if $j = j'$. Based on the choice of j we define $m+1$ sub-hybrids, which we denote by $\mathcal{H}^b_{UD,5.m+1}, \cdots \mathcal{H}^b_{UD,5.0}$. Clearly, $\mathcal{H}^0_{UD,4} = \mathcal{H}^b_{UD,5.m+1}$, $\mathcal{H}^1_{UD,4} = \mathcal{H}^b_{UD,5.0}$, and $\mathcal{H}^1_{UD,5.j} = \mathcal{H}^0_{UD,5.j+1}$. Following we prove $\mathcal{H}^0_{UD,5.j} \approx \mathcal{H}^1_{UD,5.j}$, which translates into $\mathcal{H}^0_{UD,5.j} \approx \mathcal{H}^0_{UD,5.j+1}$, based on the above, and ultimately into $\mathcal{H}^0_{UD,4} \approx \mathcal{H}^1_{UD,4}$.

Lemma 1. *Assuming the underlying FE scheme is secure as per Definition 4* $\mathcal{H}^0_{UD,5.j} \approx \mathcal{H}^1_{UD,5.j}$.

Proof. We prove this via contraposition. We construct an adversary $\mathcal{A}_{\mathsf{FE}}$ that utilizes $\mathcal{A}_{\mathsf{PAFE}}$ to win in the security game of FE. Now, assuming $\mathcal{A}_{\mathsf{PAFE}}$ issues at most m Qenc($\cdot$) queries, $\mathcal{A}_{\mathsf{FE}}$ functions as follows:

– Initialization: $\mathcal{A}_{\mathsf{FE}}$ receives mpk from $\mathcal{C}$, computes $\mathsf{pp} \leftarrow \mathsf{Com.Setup}(1^\lambda)$, samples $j^\star \leftarrow\!\!\$\ [m]$, initializes *counter* $= 0$, initializes a table $\mathcal{T}_{enc}$, samples $r_s \leftarrow \{0,1\}^\lambda$, computes $\mathsf{c}_d \leftarrow \mathsf{Com.Commit}(\top; u_d)$, samples $b' \leftarrow \{0,1\}$, and forwards the triple (pp,mpk,c_d) to $\mathcal{A}_{\mathsf{PAFE}}$.
– Encryption queries: When $\mathcal{A}_{\mathsf{PAFE}}$ issues a QEnc(x_0,x_1) query to $\mathcal{A}_{\mathsf{FE}}$, the latter issues a QEnc(x_j,x_j) query to $\mathcal{C}$ and increments *counter* by 1. $x_j = x_0$ for *counter* $< j^\star$, and $x_j = x_1$ for *counter* $> j^\star$. For *counter* $= j^\star$ $\mathcal{A}_{\mathsf{FE}}$ forwards the query to $\mathcal{C}$ without any alteration. Regardless the case, $\mathcal{C}$ returns a ciphertext ct, which $\mathcal{A}_{FE}$ forwards to $\mathcal{A}_{\mathsf{PAFE}}$.
– Functional secret key queries: When $\mathcal{A}_{\mathsf{PAFE}}$ issues a QSKeyGen query to $\mathcal{A}_{\mathsf{FE}}$, the latter forwards the query to $\mathcal{C}$, who responds with sk_f. $\mathcal{A}_{\mathsf{FE}}$ then checks if a QPKeyGen query has been issued for f. If not, it samples $r_f \leftarrow\!\!\$\ \{0,1\}^\lambda$ and computes $\mathsf{pk}_f \leftarrow \mathsf{Com.Commit}(\mathsf{sk}_f; r_f)$, $\mathcal{A}_{\mathsf{FE}}$ forwards (sk_f,pk_f) to $\mathcal{A}_{\mathsf{PAFE}}$.
– Functional public key queries: When $\mathcal{A}_{\mathsf{PAFE}}$ issues a QPKeyGen(f) query to $\mathcal{A}_{\mathsf{FE}}$, the latter checks whether $f(\mathsf{x}_{0.j^\star}) \neq f(\mathsf{x}_{1.j^\star})$. If so, $\mathcal{A}_{FE}$ samples $r_f \leftarrow\!\!\$\ \{0,1\}^\lambda$, samples $\mathsf{z}_f \leftarrow\!\!\$\ \{0,1\}^\lambda$, and computes $\mathsf{pk}_f \leftarrow \mathsf{Com.Commit}(\mathsf{z}_f; r_f)$. Otherwise, $\mathcal{A}_{FE}$ forwards a QSKeyGen(f) query to $\mathcal{C}$, who responds with sk_f. $\mathcal{A}_{FE}$ samples $r_f \leftarrow\!\!\$\ \{0,1\}^\lambda$, and computes $\mathsf{pk}_f \leftarrow \mathsf{Com.Commit}(\mathsf{sk}_f; r_f)$. In any case $\mathcal{A}_{\mathsf{FE}}$ returns pk_f to $\mathcal{A}_{\mathsf{PAFE}}$.
– Decryption queries: When $\mathcal{A}_{\mathsf{PAFE}}$ issues a QDec(ct, f) query to $\mathcal{A}_{\mathsf{FE}}$, the latter assigns $\mathsf{y} \leftarrow f(\mathsf{x}_j)$ and $\pi_d \leftarrow \mathsf{NIWI}_d.\mathsf{Prove}(\mathsf{mpk},\top,f,\perp,\perp,\mathsf{pk}_f,\mathsf{ct},\mathsf{y},\mathsf{c}_d,u_d)$. $\mathcal{A}_{\mathsf{FE}}$ forwards (y,π_d) to $\mathcal{A}_{\mathsf{PAFE}}$.

– Finalization: $\mathcal{A}_{\mathsf{PAFE}}$ outputs a bit b' which $\mathcal{A}$ forwards to $\mathcal{C}$.

The advantage $\mathcal{A}_{\mathsf{FE}}$ has in winning the FE IND-security game utilizing $\mathcal{A}_{\mathsf{PAFE}}$ is $\frac{\epsilon}{m} > \mathsf{negl}(\lambda)$. This derives from the fact that $\mathcal{A}_{\mathsf{FE}}$ needs to "guess" correctly the ct_j^b $\leftarrow$Qenc$(\cdot,\cdot)$ query for which $\mathcal{A}_{\mathsf{PAFE}}$ will issue at least one "legitimate" QDec$(\cdot$,$\mathsf{ct}_j^b)$ query; and does so by sampling $j^\star$ at random.

Thus, $\mathcal{H}^0_{UD,4} = \mathcal{H}^b_{UD,5.m+1} \approx \mathcal{H}^b_{UD,5.0} = \mathcal{H}^1_{UD,4}$ and more specifically: $Adv^{\text{Distinguish}}_{\mathcal{H}^0_{UD,4}-\mathcal{H}^1_{UD,4}}(\mathcal{A}_{\mathsf{PAFE}}) = Adv^{\text{FE-IND security}}(\mathcal{A}_{\mathsf{PAFE}})$.

Hybrid $\mathcal{H}^1_{UD,3}$: In this game we make the following change: When $\mathcal{A}_{\mathsf{PAFE}}$ issues a QPKeyGen(f) query, $\mathcal{C}$ forwards pk_f $\leftarrow$$\mathsf{PAFE.KeyGen(msk,mpk,}f)$ to $\mathcal{A}_{\mathsf{PAFE}}$. From the hiding property of the underlying commitment scheme, similarly to $\mathcal{H}^0_{UD,3} \approx \mathcal{H}^0_{UD,4}$, we get that $\mathcal{H}^1_{UD,4} \approx \mathcal{H}^1_{UD,3}$ and more specifically, $Adv^{\text{Distinguish}}_{\mathcal{H}^1_{UD,4}-\mathcal{H}^1_{UD,3}}(\mathcal{A}_{\mathsf{PAFE}}) = Adv^{\text{Com-Hidding}}(\mathcal{A}_{\mathsf{PAFE}})$.

Hybrid $\mathcal{H}^1_{UD,2}$: It is exactly the same game as $\mathcal{H}^1_{UD,3}$ except for the computation of the y. In this case, we change y to be $\mathsf{y} \leftarrow \mathsf{FE.Dec(mpk,}f\mathsf{,sk}_f\mathsf{,ct)}$, instead of $\mathsf{y} = f(\mathsf{x})$. Similarly to the case $\mathcal{H}^0_{UD,2} \approx \mathcal{H}^0_{UD,3}$, we get that $\mathcal{H}^1_{UD,3} \approx \mathcal{H}^1_{UD,2}$ and more specifically, $Adv^{\text{Distinguish}}_{\mathcal{H}^1_{UD,3}-\mathcal{H}^1_{UD,2}}(\mathcal{A}_{\mathsf{PAFE}}) = 0$.

Hybrid $\mathcal{H}^1_{UD,1}$: It is exactly the same game as $\mathcal{H}^1_{UD,2}$ except for the computation of π_d. In $\mathcal{H}^0_{UD,2}$ $\pi'_d \leftarrow \mathsf{NIWI}_d\mathsf{.Prove(mpk,}\bot\mathsf{,}f\mathsf{,}\bot\mathsf{,}\bot\mathsf{,pk}_f\mathsf{,ct,y,c}_d\mathsf{,}u_d\mathsf{)}$ using the second condition of $\mathsf{R}_{UD,d}$, whereas in $\mathcal{H}^0_{UD,1}$, using the first condition for relation $\mathsf{R}_{UD,d}$, $\pi_d \leftarrow \mathsf{NIWI}_d\mathsf{.Prove(mpk,msk,}f\mathsf{,sk}_f\mathsf{,}r_f\mathsf{,pk}_f\mathsf{,ct,y,c}_d\mathsf{,}u_d\mathsf{)}$. From the witness indistinguishability property of NIWI$_d$, similarly to $\mathcal{H}^0_{UD,1} \approx \mathcal{H}^0_{UD,2}$ we get that $\mathcal{H}^1_{UD,2} \approx \mathcal{H}^1_{UD,1}$ and more specifically, $Adv^{\text{Distinguish}}_{\mathcal{H}^1_{UD,2}-\mathcal{H}^1_{UD,1}}(\mathcal{A}_{\mathsf{PAFE}}) = Adv^{WI}_{NIWI}(\mathcal{A}_{\mathsf{PAFE}})$.

Game $\mathcal{G}^1_{UD}$: It is the security game when $b = 1$. It is exactly the same game as $\mathcal{H}^1_{UD,1}$ except for the computation of the c_d. In $\mathcal{H}^1_{UD,1}$ $\mathsf{c}'_d \leftarrow \mathsf{Com.Commit}(\top; u_d)$, whereas in $\mathcal{G}^1_{UD}$ $\mathsf{c}_d \leftarrow \mathsf{Com.Commit}(\text{msk},; u_d)$. From the hiding property of the employed commitment scheme no PPT adversary who sees a commitment can identify the committed value. Thus, $\mathcal{H}^1_{UD,1} \approx \mathcal{G}^1_{UD}$ and to be more specific, $Adv^{\text{Distinguish}}_{\mathcal{H}^1_{UD,1}-\mathcal{G}^1_{UD}}(\mathcal{A}_{\mathsf{PAFE}}) = Adv^{\text{Com-Hidding}}(\mathcal{A}_{\mathsf{PAFE}})$.

Therefore, the overall advantage $\mathcal{A}_{\mathsf{PAFE}}$ has in case *(i)*: $Adv^{\text{Distinguish}}_{\mathcal{G}^0_{UD}-\mathcal{G}^1_{UD},(i)}(\mathcal{A}_{\mathsf{PAFE}}) \leq 4 \times Adv^{\text{Com-Hidding}}(\mathcal{A}_{\mathsf{PAFE}}) + 2 \times Adv^{WI}_{NIWI}(\mathcal{A}_{\mathsf{PAFE}}) + Adv^{\text{FE-IND security}}(\mathcal{A}_{\mathsf{PAFE}})$.

Case (ii): We assume $\mathcal{A}_{\mathsf{PAFE}}$ issues no QDec$(\cdot,\cdot)$ queries and has a non-negligible advantage ϵ in winning the PAFE ecurity game. In this case we exploit the fact that $\mathcal{A}_{\mathsf{PAFE}}$ will not issue a QSKeyGen(f) query if there exists a pair of messages $(\mathsf{x}_0, \mathsf{x}_1)$ in a QEnc$(\mathsf{x}_0, \mathsf{x}_1) \rightarrow \mathsf{ct}$ query, such that $f(\mathsf{x}_0) \neq f(\mathsf{x}_1)$ and vice versa — since either way would trivially violate the winning conditions of the PAFE security game, rendering $Adv^{sec-PAFE}(\mathcal{A}_{\mathsf{PAFE}}) = 0$ $\big($see case $(\star)\big)$. We therefore can construct a "greedy" adversary $\mathcal{A}'_{\mathsf{FE}}$ who utilizes $\mathcal{A}_{\mathsf{PAFE}}$ and wins the FE IND-security game with non-negligible advantage. $\mathcal{A}'_{\mathsf{FE}}$ forwards all queries made by $\mathcal{A}_{\mathsf{PAFE}}$ to its challenger, except for QPKeyGen$(\cdot)$ ones. When $\mathcal{A}_{\mathsf{PAFE}}$ issues a QPKeyGen(f) query to $\mathcal{A}'_{\mathsf{FE}}$, the latter checks whether $\exists \mathsf{ct} \leftarrow$QEnc$(x_0, x_1)$ such that $f(\mathsf{x}_0) \neq f(\mathsf{x}_1)$. If so, $\mathcal{A}'_{\mathsf{FE}}$ samples $r_f \leftarrow\!\!\$\ \{0,1\}^\lambda$, samples $\mathsf{z}_f \leftarrow\!\!\$\ \{0,1\}^\lambda$, and computes $\mathsf{pk}_f \leftarrow \mathsf{Com.Commit}(\mathsf{z}_f; r_f)$. Otherwise, $\mathcal{A}'_{\mathsf{FE}}$ forwards a QSKeyGen(f) query to $\mathcal{C}$, who responds with sk_f. $\mathcal{A}'_{FE}$ samples $r_f \leftarrow\!\!\$\ \{0,1\}^\lambda$, and computes $\mathsf{pk}_f \leftarrow \mathsf{Com.Commit}(\mathsf{sk}_f; r_f)$. In any case $\mathcal{A}'_{\mathsf{FE}}$ returns pk_f to $\mathcal{A}_{\mathsf{PAFE}}$. Since the commitment scheme is computationally hiding $\mathcal{A}'_{\mathsf{FE}}$ has also $\epsilon > \mathsf{negl}(\lambda)$ advantage in winning the FE IND-security game, violating our initial assumption.

(Auditability). We show that no PPT adversary $\mathcal{A}_{\mathsf{PA\text{-}UD}}$ can violate the PA-UD property of PAFE, assuming a computationally sound NIWI for relation $\mathsf{R}_{UD,d}$, NIWI_d and a perfectly binding commitment scheme Com. We examine two cases. First, there is the case where the adversary $\mathcal{A}_{\mathsf{PA\text{-}UD}}$ may output a tuple T that satisfies $\mathsf{R}_{UD,d}$. If so, it either satisfies the condition that ensures that PA-UD holds $\big(\mathsf{pk}_f \leftarrow \mathsf{Com}(\mathsf{sk}_f; r_f) \wedge \mathsf{y} \leftarrow \mathsf{FE.Dec}(\mathsf{mpk}, f, \mathsf{sk}_f, \mathsf{ct})\big)$, or the "trapdoor" condition $\mathsf{c}_d \leftarrow \mathsf{Com}(\top; u_d)$. In the PA-UD setting c_d is generated by the authority (assumed to be honest in this setting), meaning that no malicious decryptor can generate a convincing proof using condition (2) of $R_{UD,d}$.

Otherwise, without loss of generality we distinguish between the following regarding the first condition: T either violates the commitment or the algorithmic condition. Since the commitment is perfectly binding, $\forall \mathsf{pk}_f \nexists (\mathsf{sk}^\star_f, r^\star_f) \neq (\mathsf{sk}_f, r_f)$ such that $\mathsf{pk}_f \leftarrow \mathsf{Com}(\mathsf{sk}^\star_f; r^\star_f) \wedge \mathsf{pk}_f \leftarrow \mathsf{Com}(\mathsf{sk}_f; r_f)$. Additionally, since mpk and ct, are provided by trusted entities and the uniquely correct sk_f is used in the FE.Dec algorithm, y is also explicitly correct (due to the correctness of the underlying FE scheme). Due to the soundness property of NIWI_d any proof $\pi^\star$ that passes verification is generated for accepting PA-UD statements using valid witnesses. Therefore, no PPT $\mathcal{A}_{\mathsf{PA\text{-}UD}}$ can break the PA-UD property with non-negligible advantage.

B Proof of Theorem 2

The proof follows in a very similar way with the one of the PA-UD construction (Theorem 1). The main difference is the inclusion of two additional hybrids for the transformation of c_e and π_e. Similarly to the previous analysis, we get the public auditability by exploiting the trusted generation of c_e, c_d, during PAFE.Setup. These commitments function as "trapdoors" for $R_{UED,e}$ and $R_{UED,d}$ respectively and allow our fabricated adversary $\mathcal{A}_{FE}$ to generate and forward convincing proofs to $\mathcal{A}_{\mathsf{PAFE}}$ without having access to encrypted values or valid functional decryption keys.

*Proof (**Security**).*

Case (i): We assume $\mathcal{A}_{\mathsf{PAFE}}$ issues at least one QDec$(\cdot,\cdot)$ query. We prove indistinguishability of the game that $\mathcal{A}_{PAFE}$ plays when $b = 0$ and $b = 1$ through a series of hybrids. Below we define the hybrids and prove them consecutively indistinguishable. The challenger bit is represented in the game/hybrid exponents.

Hybrid $\mathcal{G}^0_{UED}$: It is the security game when $b = 0$.

Hybrid $\mathcal{H}^0_{UED,1}$: It is exactly the same game as $\mathcal{G}^0_{UED}$ except for the computation of the c_d. In $\mathcal{G}^0_{UED}$ $\mathsf{c}_d \leftarrow \mathsf{Com.Commit}(\text{msk},; u_d)$, whereas in $\mathcal{H}^0_{UED,1}$ $\mathsf{c}'_d \leftarrow \mathsf{Com.Commit}(\top; u_d)$. From the hiding property of the employed commitment scheme no PPT adversary who sees a commitment can identify the committed value. Thus, $\mathcal{G}^0_{UED} \approx \mathcal{H}^0_{UED,1}$ and more specifically, $Adv^{\text{Distinguish}}_{\mathcal{G}^0_{UED}-\mathcal{H}^0_{UED,1}}(\mathcal{A}_{\mathsf{PAFE}}) = Adv^{\text{Com-Hidding}}(\mathcal{A}_{\mathsf{PAFE}})$.

Hybrid $\mathcal{H}^0_{UED,2}$: It is exactly the same game as $\mathcal{H}^0_{UED,1}$ except for the computation of the c_e. In $\mathcal{H}^0_{UED,1}$ $\mathsf{c}_e \leftarrow \mathsf{Com.Commit}(\text{msk},; u_e)$, whereas in $\mathcal{H}^0_{UED,2}$ $\mathsf{c}'_e \leftarrow \mathsf{Com.Commit}(\top; u_e)$. From the hiding property of the employed commitment scheme no PPT adversary who sees a commitment can identify the committed value. Thus, $\mathcal{H}^0_{UED,1} \approx \mathcal{H}^0_{UED,2}$ and specifically $Adv^{\text{Distinguish}}_{\mathcal{H}^0_{UED,1}-\mathcal{H}^0_{UED,2}}(\mathcal{A}_{\mathsf{PAFE}}) = Adv^{\text{Com-Hidding}}(\mathcal{A}_{\mathsf{PAFE}})$.

Hybrid $\mathcal{H}^0_{UED,3}$: It is exactly the same game as $\mathcal{H}^0_{UED,2}$ except for the computation of π_d. In $\mathcal{H}^0_{UED,2}$ $\pi_d \leftarrow \mathsf{NIWI}_d.\mathsf{Prove}(\mathsf{mpk},\mathsf{msk},f,\mathsf{sk}_f,r_f,\mathsf{pk}_f,\mathsf{ct},\mathsf{y},\mathsf{c}_d,u_d)$ using the first condition for relation $\mathsf{R}_{UED,d}$, whereas in $\mathcal{H}^0_{UED,3}$, using the second condition of $\mathsf{R}_{UED,d}$, $\pi'_d \leftarrow \mathsf{NIWI}_d.\mathsf{Prove}(\mathsf{mpk},\perp,f,\perp,\perp,\mathsf{pk}_f,\mathsf{ct},\mathsf{y},\mathsf{c}_d,u_d)$ respectively. From the witness indistinguishability property of NIWI$_d$ no PPT adversary can distinguish between which condition is satisfied for the generation of π_d. Thus, $\mathcal{H}^0_{UED,2} \approx \mathcal{H}^0_{UED,3}$ and more specifically, $Adv^{\text{Distinguish}}_{\mathcal{H}^0_{UED,2}-\mathcal{H}^0_{UED,3}}(\mathcal{A}_{\mathsf{PAFE}}) = Adv^{WI}_{NIWI}(\mathcal{A}_{\mathsf{PAFE}})$.

Hybrid $\mathcal{H}^0_{UED,4}$: It is exactly the same game as $\mathcal{H}^0_{UED,3}$ except for the computation of π_e. In $\mathcal{H}^0_{UED,3}$ $\pi_e \leftarrow \mathsf{NIWI}_e.\mathsf{Prove}(\mathsf{mpk},\mathsf{msk},\mathsf{ek},\mathsf{x},\mathsf{ct},\mathsf{c}_e,u_e,s_e)$ using the first condition for relation $\mathsf{R}_{UED,e}$, whereas in $\mathcal{H}^0_{UED,4}$, using the second condition of $\mathsf{R}_{UED,e}$, $\pi'_e \leftarrow \mathsf{NIWI}_d.\mathsf{Prove}(\mathsf{mpk},\perp,\perp,\perp,\mathsf{ct},\mathsf{c}_e,u_e,\perp)$ respectively. From the witness indistinguishability property of NIWI$_e$ no PPT adversary can distinguish between which condition is satisfied for the generation of π_e. Thus, $\mathcal{H}^0_{UED,3} \approx \mathcal{H}^0_{UED,4}$ and more specifically, $Adv^{\text{Distinguish}}_{\mathcal{H}^0_{UED,3}-\mathcal{H}^0_{UED,4}}(\mathcal{A}_{\mathsf{PAFE}}) = Adv^{WI}_{NIWI}(\mathcal{A}_{\mathsf{PAFE}})$.

Hybrid $\mathcal{H}^0_{UED,5}$: It is exactly the same game as $\mathcal{H}^0_{UED,6}$ except for the computation of the y. In this case, we change y to be $\mathsf{y} = f(\mathsf{x})$ instead of $\mathsf{y} \leftarrow \mathsf{FE.Dec}(\mathsf{mpk},f,\mathsf{sk}_f,\mathsf{ct})$. Remember that for $\mathcal{A}_{\mathsf{PAFE}}$ to have non-negligible chance of winning in its game, it must be that for all functions f that $\mathcal{A}_{\mathsf{PAFE}}$ issues a QSKeyGen(f) query, for all $\mathsf{ct} \leftarrow$ QEnc(x_0,x_1): $f(\mathsf{x}) = f(\mathsf{x}_0) = f(\mathsf{x}_1)$. Additionally and similarly, for all functions f for which $\mathcal{A}_{\mathsf{PAFE}}$ has issued QPKeyGen(f) and QDec(ct,f) queries, where $\mathsf{ct} \leftarrow$ QEnc(x_0,x_1), it must be that $f(\mathsf{x}) = f(\mathsf{x}_0) = f(\mathsf{x}_1)$. In any other case by the restrictions of the security game for PAFE $Adv^{sec-PAFE}(\mathcal{A}_{\mathsf{PAFE}}(1^\lambda)) = 0$. Since $\mathcal{A}_{\mathsf{PAFE}}$ cannot win in any of these two games with non-negligible advantage unless $f(\mathsf{x}_0) = f(\mathsf{x}_1)$, $\mathcal{H}^0_{UED,4} \approx \mathcal{H}^0_{UED,5}$ and more specifically, $Adv^{\text{Distinguish}}_{\mathcal{H}^0_{UED,4}-\mathcal{H}^0_{UED,5}}(\mathcal{A}_{\mathsf{PAFE}}) = 0$.

Hybrid $\mathcal{H}^0_{UED,6}$: In this game we make the following change: the challenger samples $j' \leftarrow\$ \{0, \cdots, m+1\}$, initializes a counter $j = 0$, and when $\mathcal{A}_{\mathsf{PAFE}}$ issues an encryption query, the challenger sets $j = j + 1$ and returns ct^b_j (we denote that query as QEnc($\mathsf{x}_{0,j}$,$\mathsf{x}_{1,j}$), more concretely). Now, when $\mathcal{A}_{\mathsf{PAFE}}$ issues a QPKeyGen(f) query, $\mathcal{C}$ checks whether $f(\mathsf{x}_{0,j'}) \neq f(\mathsf{x}_{1,j'})$. If so, it samples z_f, $r_f \leftarrow\$ \mathbb{Z}_p$ and computes $\mathsf{pk}_f \leftarrow \mathsf{Com.Commit}(\mathsf{z}_f,r_f)$. Remember that since $f(\mathsf{x}_{0,j'}) \neq f(\mathsf{x}_{1,j'})$ the adversary cannot issue a QSKeyGen(f) or a QDec($\mathsf{ct}^b_{j'}$,f) query — for that particular ciphertext. In such cases $\mathcal{A}_{\mathsf{PAFE}}$ would trivially diminish its advantage to 0, contradicting our assumption that it has non-negligible advantage ϵ in winning the security game for PAFE. Therefore, from the hiding property of the underlying commitment scheme, similarly to $\mathcal{G}^0_{UED} \approx \mathcal{H}_{UED,1}$, we get that $\mathcal{H}^0_{UED,5} \approx \mathcal{H}^0_{UED,6}$ and more specifically, $Adv^{\text{Distinguish}}_{\mathcal{H}^0_{UED,5}-\mathcal{H}^0_{UED,6}}(\mathcal{A}_{\mathsf{PAFE}}) = Adv^{\text{Com-Hidding}}(\mathcal{A}_{\mathsf{PAFE}})$.

Hybrid $\mathcal{H}^b_{UD,7.j}$: We now define a series of hybrids, indexed by j. In these hybrids we make the following change: the challenger $\mathcal{C}$ samples $b \leftarrow\$ \{0,1\}$ and when $\mathcal{A}_{\mathsf{PAFE}}$ issues a QEnc(x_0,x_1) query $\mathcal{C}$ returns $\mathsf{ct}^0 \leftarrow \mathsf{PAFE.Enc}(\mathsf{mpk},\mathsf{ek},\mathsf{x}_0)$, if $j < j'$, $\mathsf{ct}^1 \leftarrow \mathsf{PAFE.Enc}(\mathsf{mpk},\mathsf{ek},\mathsf{x}_1)$, if $j > j'$, and $\mathsf{ct}^b \leftarrow \mathsf{PAFE.Enc}(\mathsf{mpk},\mathsf{ek},\mathsf{x}_b)$, if $j = j'$. Based on the choice of j we define $m+1$ sub-hybrids, which we denote by $\mathcal{H}^b_{UED,7.m+1}, \cdots \mathcal{H}^b_{UED,7.0}$. Clearly, $\mathcal{H}^0_{UED,6} = \mathcal{H}^b_{UED,7.m+1}$, $\mathcal{H}^1_{UD,6} = \mathcal{H}^b_{UED,7.0}$, and $\mathcal{H}^1_{UED,7.j} = \mathcal{H}^0_{UED,7.j+1}$. Following we prove $\mathcal{H}^0_{UED,7.j} \approx \mathcal{H}^1_{UED,7.j}$, which translates into $\mathcal{H}^0_{UED,7.j} \approx \mathcal{H}^0_{UED,7.j+1}$, based on the above, and ultimately into $\mathcal{H}^0_{UED,6} \approx \mathcal{H}^1_{UED,6}$.

Lemma 2. *Assuming the underlying FE scheme is secure as per Definition 4* $\mathcal{H}^0_{UED,7.j} \approx \mathcal{H}^1_{UED,7.j}$.

Proof. We prove this via contraposition. We construct an adversary $\mathcal{A}_{\mathsf{FE}}$ that utilizes $\mathcal{A}_{\mathsf{PAFE}}$ to win in the security game of FE. Now, assuming $\mathcal{A}_{\mathsf{PAFE}}$ issues at most m Qenc($\cdot$) queries, $\mathcal{A}_{\mathsf{FE}}$ functions as follows:

– Initialization: $\mathcal{A}_{\mathsf{FE}}$ receives mpk from $\mathcal{C}$, computes $\mathsf{pp} \leftarrow \mathsf{Com.Setup}(1^\lambda)$, samples $j^\star \leftarrow\$ [m]$, initializes *counter* $= 0$, initializes a table $\mathcal{T}_{enc}$, samples $r_s \leftarrow \{0,1\}^\lambda$, computes $\mathsf{c}_d \leftarrow \mathsf{Com.Commit}(\top; u_d)$, samples $b' \leftarrow \{0,1\}$, and forwards the triple (pp,mpk,c_d) to $\mathcal{A}_{\mathsf{PAFE}}$.
– Encryption queries: When $\mathcal{A}_{\mathsf{PAFE}}$ issues a QEnc(x_0,x_1) query to $\mathcal{A}_{\mathsf{FE}}$, the latter issues a QEnc(x_j,x_j) query to $\mathcal{C}$ and increments *counter* by 1. $x_j = x_0$ for *counter* $< j^\star$, and $x_j = x_1$ for *counter* $> j^\star$. For *counter* $= j^\star$ $\mathcal{A}_{\mathsf{FE}}$ forwards the query to $\mathcal{C}$ without any alteration. Also, $\mathcal{A}_{FE}$ computes $\pi_e \leftarrow \mathsf{NIWI}_d.\mathsf{Prove}(\mathsf{mpk},\perp,\perp,\perp,\mathsf{ct},\mathsf{c}_e,u_e,\perp)$. Regardless the case, $\mathcal{C}$ returns a ciphertext ct and $\mathcal{A}_{FE}$ forwards (ct,π_e) to $\mathcal{A}_{\mathsf{PAFE}}$.
– Functional secret key queries: When $\mathcal{A}_{\mathsf{PAFE}}$ issues a QSKeyGen query to $\mathcal{A}_{\mathsf{FE}}$, the latter forwards the query to $\mathcal{C}$, who responds with sk_f. $\mathcal{A}_{\mathsf{FE}}$ then checks if a QPKeyGen query has been issued for f. If not, it samples $r_f \leftarrow\$ \{0,1\}^\lambda$ and computes $\mathsf{pk}_f \leftarrow \mathsf{Com.Commit}(\mathsf{sk}_f; r_f)$, $\mathcal{A}_{\mathsf{FE}}$ forwards (sk_f,pk_f) to $\mathcal{A}_{\mathsf{PAFE}}$.

– Functional public key queries: When $\mathcal{A}_{\mathsf{PAFE}}$ issues a QPKeyGen(f) query to $\mathcal{A}_{\mathsf{FE}}$, the latter checks whether $f(\mathsf{x}_{0.j^\star}) \neq f(\mathsf{x}_{1.j^\star})$. If so, $\mathcal{A}_{FE}$ samples $r_f \leftarrow\!\$ \{0,1\}^\lambda$, samples $\mathsf{z}_f \leftarrow\!\$ \{0,1\}^\lambda$, and computes $\mathsf{pk}_f \leftarrow \mathsf{Com.Commit}(\mathsf{z}_f; r_f)$. Otherwise, $\mathcal{A}_{FE}$ forwards a QSKeyGen(f) query to $\mathcal{C}$, who responds with sk_f. $\mathcal{A}_{FE}$ samples $r_f \leftarrow\!\$ \{0,1\}^\lambda$, and computes $\mathsf{pk}_f \leftarrow \mathsf{Com.Commit}(\mathsf{sk}_f; r_f)$. In any case $\mathcal{A}_{\mathsf{FE}}$ returns pk_f to $\mathcal{A}_{\mathsf{PAFE}}$.
– Decryption queries: When $\mathcal{A}_{\mathsf{PAFE}}$ issues a QDec(ct, f) query to $\mathcal{A}_{\mathsf{FE}}$, the latter assigns $\mathsf{y} \leftarrow f(\mathsf{x}_j)$ and $\pi_d \leftarrow \mathsf{NIWI}_d.\mathsf{Prove}(\mathsf{mpk},\top,f,\bot,\bot,\mathsf{pk}_f,\mathsf{ct},\mathsf{y},\mathsf{c}_d,u_d)$. $\mathcal{A}_{\mathsf{FE}}$ forwards (y,π_d) to $\mathcal{A}_{\mathsf{PAFE}}$.
– Finalization: $\mathcal{A}_{\mathsf{PAFE}}$ outputs a bit b' which $\mathcal{A}$ forwards to $\mathcal{C}$.

The advantage $\mathcal{A}_{\mathsf{FE}}$ has in winning the FE IND-security game utilizing $\mathcal{A}_{\mathsf{PAFE}}$ is $\frac{\epsilon}{m} > \mathsf{negl}(\lambda)$. This derives from the fact that $\mathcal{A}_{\mathsf{FE}}$ needs to "guess" correctly the $\mathsf{ct}_j^b \leftarrow$Qenc$(\cdot,\cdot)$ query for which $\mathcal{A}_{\mathsf{PAFE}}$ will issue at least one "legitimate" QDec($\cdot$,ct_j^b) query; and does so by sampling $j^\star$ at random.

Thus, $\mathcal{H}^0_{UED,6} = \mathcal{H}^b_{UED,7.m+1} \approx \mathcal{H}^b_{UED,7.0} = \mathcal{H}^1_{UED,6}$ and more specifically: $Adv^{\text{Distinguish}}_{\mathcal{H}^0_{UED,6}-\mathcal{H}^1_{UED,6}}(\mathcal{A}_{\mathsf{PAFE}}) = Adv^{\text{FE-IND security}}(\mathcal{A}_{\mathsf{PAFE}})$.

Hybrid $\mathcal{H}^1_{UD,5}$: In this game we make the following change: When $\mathcal{A}_{\mathsf{PAFE}}$ issues a QPKeyGen(f) query, $\mathcal{C}$ forwards $\mathsf{pk}_f \leftarrow$PAFE.KeyGen(msk,mpk,f) to $\mathcal{A}_{\mathsf{PAFE}}$. From the hiding property of the underlying commitment scheme, similarly to $\mathcal{H}^0_{UED,5} \approx \mathcal{H}^0_{UED,6}$, we get that $\mathcal{H}^1_{UED,6} \approx \mathcal{H}^1_{UED,5}$ and more specifically, $Adv^{\text{Distinguish}}_{\mathcal{H}^1_{UED,6}-\mathcal{H}^1_{UED,5}}(\mathcal{A}_{\mathsf{PAFE}}) = Adv^{\text{Com-Hidding}}(\mathcal{A}_{\mathsf{PAFE}})$.

Hybrid $\mathcal{H}^1_{UED,4}$: It is exactly the same game as $\mathcal{H}^1_{UED,5}$ except for the computation of the y. In this case, we change y to be $\mathsf{y} \leftarrow \mathsf{FE.Dec}(\mathsf{mpk},f,\mathsf{sk}_f,\mathsf{ct})$, instead of $\mathsf{y} = f(\mathsf{x})$. Similarly to the case $\mathcal{H}^0_{UED,4} \approx \mathcal{H}^0_{UED,5}$, we get that $\mathcal{H}^1_{UED,5} \approx \mathcal{H}^1_{UED,4}$ and more specifically, $Adv^{\text{Distinguish}}_{\mathcal{H}^1_{UED,5}-\mathcal{H}^1_{UED,4}}(\mathcal{A}_{\mathsf{PAFE}}) = 0$.

Hybrid $\mathcal{H}^0_{UED,3}$: It is exactly the same game as $\mathcal{H}^0_{UED,4}$ except for the computation of π_e. In $\mathcal{H}^0_{UED,4}$ $\pi_e \leftarrow \mathsf{NIWI}_d.\mathsf{Prove}(\mathsf{mpk},\bot,\bot,\bot,\mathsf{ct},\mathsf{c}_e,u_e,\bot)$ using the second condition of $\mathsf{R}_{UED,e}$, whereas in $\mathcal{H}^0_{UED,3}$, using the first condition for relation $\mathsf{R}_{UED,e}$, $\pi_e \leftarrow \mathsf{NIWI}_d.\mathsf{Prove}(\mathsf{mpk},\mathsf{msk},\mathsf{ek},\mathsf{x},\mathsf{ct},\mathsf{c}_e,u_e,s_e)$. From the witness indistinguishability property of NIWI$_e$, similarly to $\mathcal{H}^0_{UED,3} \approx \mathcal{H}^0_{UED,4}$ we get that $\mathcal{H}^1_{UED,4} \approx \mathcal{H}^1_{UED,3}$ and more specifically, $Adv^{\text{Distinguish}}_{\mathcal{H}^1_{UED,4}-\mathcal{H}^1_{UED,3}}(\mathcal{A}_{\mathsf{PAFE}}) = Adv^{WI}_{NIWI}(\mathcal{A}_{\mathsf{PAFE}})$.

Hybrid $\mathcal{H}^0_{UED,2}$: It is exactly the same game as $\mathcal{H}^0_{UED,3}$ except for the computation of π_d. In $\mathcal{H}^0_{UED,3}$ $\pi'_d \leftarrow \mathsf{NIWI}_d.\mathsf{Prove}(\mathsf{mpk},\bot,f,\bot,\bot,\mathsf{pk}_f,\mathsf{ct},\mathsf{y},\mathsf{c}_d,u_d)$ using the second condition of $\mathsf{R}_{UD,d}$, whereas in $\mathcal{H}^0_{UED,2}$, using the first condition for relation $\mathsf{R}_{UD,d}$, $\pi_d \leftarrow \mathsf{NIWI}_d.\mathsf{Prove}(\mathsf{mpk},\mathsf{msk},f,\mathsf{sk}_f,r_f,\mathsf{pk}_f,\mathsf{ct},\mathsf{y},\mathsf{c}_d,u_d)$. From the witness indistinguishability property of NIWI$_d$, similarly to $\mathcal{H}^0_{UED,2} \approx \mathcal{H}^0_{UED,3}$ we get that $\mathcal{H}^1_{UED,3} \approx \mathcal{H}^1_{UED,2}$ and more specifically, $Adv^{\text{Distinguish}}_{\mathcal{H}^1_{UED,3}-\mathcal{H}^1_{UED,2}}(\mathcal{A}_{\mathsf{PAFE}}) = Adv^{WI}_{NIWI}(\mathcal{A}_{\mathsf{PAFE}})$.

Game $\mathcal{H}^1_{UED,1}$: It is exactly the same game as $\mathcal{H}^1_{UED,2}$ except for the computation of the c_e. In $\mathcal{H}^1_{UED,1}$ $\mathsf{c}'_e \leftarrow \mathsf{Com.Commit}(\top; u_e)$, whereas in $\mathcal{H}^1_{UED,1}$ $\mathsf{c}_e \leftarrow \mathsf{Com.Commit}(\text{msk},; u_e)$. From the hiding property of the employed commitment scheme no PPT adversary who sees a commitment can identify the committed value. Thus, $\mathcal{H}^1_{UED,2} \approx \mathcal{H}^1_{UED,1}$ and specifically, $Adv^{\text{Distinguish}}_{\mathcal{H}^1_{UED,2}-\mathcal{H}^1_{UED,1}}(\mathcal{A}_{\mathsf{PAFE}}) = Adv^{\text{Com-Hidding}}(\mathcal{A}_{\mathsf{PAFE}})$.

Game $\mathcal{G}^1_{UED}$: It is the security game when $b = 1$. It is exactly the same game as $\mathcal{H}^1_{UED,1}$ except for the computation of the c_d. In $\mathcal{H}^1_{UED,1}$ the commitment c'_d $\leftarrow \mathsf{Com.Commit}(\top; u_d)$, whereas in $\mathcal{G}^1_{UED}$ $\mathsf{c}_d \leftarrow \mathsf{Com.Commit}(\text{msk}; u_d)$. From the hiding property of the employed commitment scheme no PPT adversary who sees a commitment can identify the committed value. Thus, $\mathcal{H}^1_{UED,1} \approx \mathcal{G}^1_{UED}$ and more specifically, $Adv^{\text{Distinguish}}_{\mathcal{H}^1_{UED,1}-\mathcal{G}^1_{UED}}(\mathcal{A}_{\mathsf{PAFE}}) = Adv^{\text{Com-Hidding}}(\mathcal{A}_{\mathsf{PAFE}})$.

Thus, the overall advantage $\mathcal{A}_{\mathsf{PAFE}}$ has in case *(i)*: $Adv^{\text{Distinguish}}_{\mathcal{G}^0_{UED}-\mathcal{G}^1_{UED},(i)}(\mathcal{A}_{\mathsf{PAFE}}) \leq 6 \times Adv^{\text{Com-Hidding}}(\mathcal{A}_{\mathsf{PAFE}}) + 4 \times Adv^{WI}_{NIWI}(\mathcal{A}_{\mathsf{PAFE}}) + Adv^{\text{FE-IND security}}(\mathcal{A}_{\mathsf{PAFE}})$.

Case (ii): We assume $\mathcal{A}_{\mathsf{PAFE}}$ issues no QDec$(\cdot,\cdot)$ queries and has a non-negligible advantage ϵ in winning the PAFE ecurity game. In this case we exploit the fact that $\mathcal{A}_{\mathsf{PAFE}}$ will not issue a QSKeyGen(f) query if there exists a pair of messages $(\mathsf{x}_0, \mathsf{x}_1)$ in a QEnc$(\mathsf{x}_0, \mathsf{x}_1) \rightarrow \mathsf{ct}$ query, such that $f(\mathsf{x}_0) \neq f(\mathsf{x}_1)$ and vice versa — since either way would trivially violate the winning conditions of the PAFE security game, rendering $Adv^{sec-PAFE}(\mathcal{A}_{\mathsf{PAFE}}) = 0$ $\big($see case $(^\star)\big)$. We therefore can construct a "greedy" adversary $\mathcal{A}'_{\mathsf{FE}}$ who utilizes $\mathcal{A}_{\mathsf{PAFE}}$ and wins the FE IND-security game with non-negligible advantage. $\mathcal{A}'_{\mathsf{FE}}$ forwards all queries made by $\mathcal{A}_{\mathsf{PAFE}}$ to its challenger, except for QPKeyGen$(\cdot)$ ones. When $\mathcal{A}_{\mathsf{PAFE}}$ issues a QPKeyGen(f) query to $\mathcal{A}'_{\mathsf{FE}}$, the latter checks whether $\exists \mathsf{ct} \leftarrow$QEnc$(x_0, x_1)$ such that $f(\mathsf{x}_0) \neq f(\mathsf{x}_1)$. If so, $\mathcal{A}'_{\mathsf{FE}}$ samples $r_f \leftarrow\!\$\ \{0,1\}^\lambda$, samples $\mathsf{z}_f \leftarrow\!\$\ \{0,1\}^\lambda$, and computes $\mathsf{pk}_f \leftarrow \mathsf{Com.Commit}(\mathsf{z}_f; r_f)$. Otherwise, $\mathcal{A}'_{\mathsf{FE}}$ forwards a QSKeyGen(f) query to $\mathcal{C}$, who responds with sk_f. $\mathcal{A}'_{FE}$ samples $r_f \leftarrow\!\$\ \{0,1\}^\lambda$, and computes $\mathsf{pk}_f \leftarrow \mathsf{Com.Commit}(\mathsf{sk}_f; r_f)$. In any case $\mathcal{A}'_{\mathsf{FE}}$ returns pk_f to $\mathcal{A}_{\mathsf{PAFE}}$. Since the commitment scheme is computationally hiding $\mathcal{A}'_{\mathsf{FE}}$ has also $\epsilon > \mathsf{negl}(\lambda)$ advantage in winning the FE IND-security game, violating our initial assumption.

(Auditability). We show that no PPT adversary $\mathcal{A}_{\mathsf{PA\text{-}UED}}$ can violate the PA-UED auditability property of PAFE, assuming a computationally sound NIWI for relation $\mathsf{R}_{UED,d}$, NIWI_d, a computationally sound NIWI for relation $\mathsf{R}_{UED,e}$, NIWI_e, and a perfectly binding commitment scheme Com.

We examine two cases. First, the case where the adversary $\mathcal{A}_{\mathsf{PA\text{-}UED}}$ may output a tuple $\mathtt{T}$ that satisfies $\mathsf{R}_{UED,d}$. If so, it either satisfies the condition that ensures that PA-UED holds $(\mathsf{pk}_f \leftarrow \mathsf{Com}(\mathsf{sk}_f; r_f) \wedge \mathsf{y} \leftarrow \mathsf{FE.Dec}(\mathsf{mpk}, f, \mathsf{sk}_f, \mathsf{ct}))$, or the "trapdoor" condition $\mathsf{c}_s \leftarrow \mathsf{Com}(0^{len}; r_s)$. In the PA-UED setting c_d is generated by a trusted authority, meaning that no malicious decryptor can generate a convincing proof using condition (2) of $R_{UED,d}$. Similarly, no PPT adversary can provide a convincing proof that satisfies the second condition of $\mathsf{R}_{UED,e}$.

Otherwise, without loss of generality we distinguish between the following: $\mathtt{T}$ either violates any of the commitments or the algorithmic conditions. Since the commitment is perfectly binding, $\forall \mathsf{pk}_f \nexists (\mathsf{sk}^\star_f, r^\star_f) \neq (\mathsf{sk}_f, r_f)$ such that $\mathsf{pk}_f \leftarrow \mathsf{Com}(\mathsf{sk}^\star_f; r^\star_f) \wedge \mathsf{pk}_f \leftarrow \mathsf{Com}(\mathsf{sk}_f; r_f)$. Additionally, since mpk, f, and ct are provided by the trusted authority and the uniquely correct sk_f is used in the $\mathsf{FE.Dec}$ algorithm, y is also explicitly correct (due to the correctness of the underlying FE scheme). Due to the soundness property of NIWI_d any proof $\pi^\star_d$ that passes verification is generated for accepting $\mathsf{PA\text{-}UED}$ statements using valid witnesses. Similarly, regarding $\mathsf{R}_{UED,e}$, c_e is generated in a trusted manner and due to the soundness property of NIWI_e any proof $\pi^\star_e$ that passes verification is generated for accepting $\mathsf{PA\text{-}UED}$ statements using valid witnesses. Therefore, no PPT $\mathcal{A}_{\mathsf{PA\text{-}UED}}$ can break the $\mathsf{PA\text{-}UED}$ property with non-negligible advantage.

C Proof of Theorem 3

*Proof (**Security**).* To prove our construction secure we show that the execution of the security game when the challenger chooses $b = 0$ is indistinguishable from the one when it picks $b = 1$. The proof comprises of a series of proofs that render the hybrids below (which are depicted in Table 2) indistinguishable, consecutively. Symbol $'$ signifies the elements that are not used in the generation of proofs π_d, π_f respectively.

- Hybrid $\mathcal{H}_{UAD,0}$: This is the security game with $b = 0$. The master public key is $\mathsf{mpk}' = \{\mathsf{mpk}_i\}_{i\in[4]}$, where $\forall i \in [4]$ $(\mathsf{msk}_i, \mathsf{mpk}_i, \mathsf{ek}_i) \leftarrow \mathsf{Setup}(1^\lambda; s_i)$ for some random string s_i. The challenge ciphertext is $\mathsf{ct}' \leftarrow \{\mathsf{ct}_i\}_{i\in[4]}$, where $\forall i \in [4]$ $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{ek}_i, \mathsf{x}_0; \mathsf{r}_{e,i})$, for some random string $\mathsf{r}_{e,i}$. The functional secret keys are $\mathsf{sk}'_f = \{\mathsf{sk}_{f,i}, r_{f,i}\}_{i\in[4]}$, where $\forall i \in [4]$ and for random strings $\mathsf{r}_{f,i}$ and $\mathsf{r}_{k,i}$, $\mathsf{sk}_{f,i} \leftarrow \mathsf{FE.KeyGen}(\mathsf{mpk}_i, \mathsf{msk}_i, f_i; r_{k_i})$. The functional public keys are $\mathsf{pk}_f = \{\mathsf{pk}_{f,i}\}_{i\in[4]}$, where $\forall i \in [4]$ $\mathsf{pk}_{f,i} \leftarrow \mathsf{Com}(\mathsf{sk}_{f,i}; r_{f,i})$. The functional output $\mathsf{y}_0 \leftarrow \mathsf{Dec}(\mathsf{mpk}_i, f, \mathsf{sk}_{f,i}, \mathsf{ct}_i)$, $\forall i \in [4]$. π_f is computed for condition (1) of relation $\mathsf{R}_{UAD,f}$ and π_d is computed for condition (1) of relation $\mathsf{R}_{UAD,d}$.
- Hybrid $\mathcal{H}_{UAD,1}$: This hybrid is identical to the previous one, except for the computation of π_d. Using indices $\{1, 2\}$, π_d is computed for condition (2) of relation $\mathsf{R}_{UAD,d}$.
- Hybrid $\mathcal{H}_{UAD,2}$: This hybrid is identical to the previous one, except for the computation of π_f. Using indices $\{1, 2, 3\}$, π_f is computed for condition (2) of relation $\mathsf{R}_{UAD,f}$.
- Hybrid $\mathcal{H}_{UAD,3}$: This hybrid is identical to the previous one, except for the computation of ct_4, i.e., $\mathsf{ct}_4 \leftarrow \mathsf{Enc}(\mathsf{ek}_4, \mathsf{x}_1; \mathsf{r}_{e,4})$.
- Hybrid $\mathcal{H}_{UAD,4}$: This hybrid is identical to the previous one, except for the computation of π_f, which is computed using indices $\{1, 2, 4\}$.
- Hybrid $\mathcal{H}_{UAD,5}$: This hybrid is identical to the previous one, except for ct_3, i.e., $\mathsf{ct}_3 \leftarrow \mathsf{Enc}(\mathsf{ek}_3, \mathsf{x}_1; \mathsf{r}_{e,3})$.
- Hybrid $\mathcal{H}_{UAD,6}$: This hybrid is identical to the previous one, except for the computation of π_d, which is computed using indices $\{1, 4\}$.

$\mathcal{H}_{UAD}$	$\{\mathsf{ct}_1,\mathsf{ct}_2,\mathsf{ct}_3,\mathsf{ct}_4\}$	$\{(\mathsf{sk}_f,\mathsf{pk}_f)_i\}_i$	π_f	π_d	Security
0	$\mathsf{x}_0,\mathsf{x}_0,\mathsf{x}_0,\mathsf{x}_0$	f,f,f,f	(1)	(1)	-
1	$\mathsf{x}_0,\mathsf{x}_0,\mathsf{x}_0',\mathsf{x}_0'$	f,f,f,f	(1)	(2)	NIWI$_d$
2	$\mathsf{x}_0,\mathsf{x}_0,\mathsf{x}_0',\mathsf{x}_0'$	f,f,f,f'	(2)	(2)	NIWI$_f$
3	$\mathsf{x}_0,\mathsf{x}_0,\mathsf{x}_0',\mathsf{x}_1'$	f,f,f,f'	(2)	(2)	FE
4	$\mathsf{x}_0,\mathsf{x}_0,\mathsf{x}_0',\mathsf{x}_1'$	f,f,f',f	(2)	(2)	NIWI$_f$
5	$\mathsf{x}_0,\mathsf{x}_0,\mathsf{x}_1',\mathsf{x}_1'$	f,f,f',f	(2)	(2)	FE
6	$\mathsf{x}_0,\mathsf{x}_0',\mathsf{x}_1',\mathsf{x}_1$	f,f,f',f	(2)	(2)	NIWI$_d$
7	$\mathsf{x}_0,\mathsf{x}_0',\mathsf{x}_1',\mathsf{x}_1$	f,f',f,f	(2)	(2)	NIWI$_f$
8	$\mathsf{x}_0,\mathsf{x}_1',\mathsf{x}_1',\mathsf{x}_1$	f,f',f,f	(2)	(2)	FE
9	$\mathsf{x}_0',\mathsf{x}_1',\mathsf{x}_1,\mathsf{x}_1$	f,f',f,f	(2)	(2)	NIWI$_d$
10	$\mathsf{x}_0',\mathsf{x}_1',\mathsf{x}_1,\mathsf{x}_1$	f',f,f,f	(2)	(2)	NIWI$_f$
11	$\mathsf{x}_1',\mathsf{x}_1',\mathsf{x}_1,\mathsf{x}_1$	f',f,f,f	(2)	(2)	FE
12	$\mathsf{x}_1',\mathsf{x}_1',\mathsf{x}_1,\mathsf{x}_1$	f,f,f,f	(1)	(2)	NIWI$_f$
13	$\mathsf{x}_1,\mathsf{x}_1,\mathsf{x}_1,\mathsf{x}_1$	f,f,f,f	(1)	(1)	NIWI$_d$

Table 2. Hybrids for the security proof of the PA-UAD PAFE.

- Hybrid $\mathcal{H}_{UAD,7}$: This hybrid is identical to the previous one, except for the computation of π_f, which is computed using indices $\{1,3,4\}$.
- Hybrid $\mathcal{H}_{UAD,8}$: This hybrid is identical to the previous one, except for the computation of ct_2, i.e., $\mathsf{ct}_2 \leftarrow \mathsf{Enc}(\mathsf{ek}_2,\mathsf{x}_1;\mathsf{r}_{e,2})$.
- Hybrid $\mathcal{H}_{UAD,9}$: This hybrid is identical to the previous one, except for the computation of π_d, which is computed using indices $\{3,4\}$.
- Hybrid $\mathcal{H}_{UAD,10}$: This hybrid is identical to the previous one, except for the computation of π_f, which is computed using indices $\{2,3,4\}$.
- Hybrid $\mathcal{H}_{UAD,11}$: This hybrid is identical to the previous one, except for ct_1, i.e., $\mathsf{ct}_1 \leftarrow \mathsf{Enc}(\mathsf{ek}_1,\mathsf{x}_1;\mathsf{r}_{e,1})$.
- Hybrid $\mathcal{H}_{UAD,12}$: This hybrid is identical to the previous one, except for the computation of π_f. Using indices $\{1,2,3,4\}$, π_f is computed for condition (1) of relation $\mathsf{R}_{UAD,f}$.
- Hybrid $\mathcal{H}_{UAD,13}$: This hybrid is identical to the previous one, except for the computation of π_d. Using indices $\{1,2,3,4\}$, π_d is computed for condition (1) of relation $\mathsf{R}_{UAD,d}$. Additionally, it is the real game with challenger bit b=1.

[$\mathcal{H}_{UAD,0} \approx \mathcal{H}_{UAD,1}$] Assuming that NIWI_d is a non-interactive witness indistinguishable proof system, the outputs of experiments as described in $\mathcal{H}_{UAD,0}$ and $\mathcal{H}_{UAD,1}$ are computationally indistinguishable.

Proof. The only difference between the two hybrids is the manner in which proof π_d is computed. In $\mathcal{H}_{UAD,0}$ is computed for condition (1) of $\mathsf{R}_{UAD,d}$ using the real witness using indices $\{1,2,3,4\}$, whereas in $\mathcal{H}_{UAD,1}$ is computed for condition (2) of $\mathsf{R}_{UAD,d}$ using the indices $\{1,2\}$. We can construct an adversary that plays the witness indistinguishability game for NIWI$_d$, and internally acts as the challenger for the indistinguishability game of PAFE. The advantage of the NIWI-adversary is bound by the PAFE-adversary, since NIWI_d is a non-interactive witness indistinguishable proof system, $\mathcal{H}_{UAD,0}$ and $\mathcal{H}_{UAD,1}$ are computationally indistinguishable.

[$\mathcal{H}_{UAD,1} \approx \mathcal{H}_{UAD,2}$] Assuming that NIWI_f is a non-interactive witness indistinguishable proof system, the outputs of experiments as described in $\mathcal{H}_{UAD,1}$ and $\mathcal{H}_{UAD,2}$ are computationally indistinguishable.

Proof. The only difference between the two hybrids is the manner in which proof π_f is computed. In $\mathcal{H}_{UAD,1}$ is computed for condition (1) of $\mathsf{R}_{UAD,f}$ using the "real" witness for indices $\{1, 2, 3, 4\}$, whereas in $\mathcal{H}_{UAD,2}$ is computed for condition (2) of $\mathsf{R}_{UAD,f}$ using the indices $\{1, 2, 3\}$. Similarly to the previous analysis we can construct an adversary that plays the witness indistinguishability game for NIWI$_f$, and internally acts as the challenger for the indistinguishability game of PAFE. The advantage of the NIWI-adversary is bound by the PAFE-adversary, and since NIWI_f is a non-interactive witness indistinguishable proof system, $\mathcal{H}_{UAD,1}$ and $\mathcal{H}_{UAD,2}$ are computationally indistinguishable.

[$\mathcal{H}_{UAD,2} \approx \mathcal{H}_{UAD,3}$] Assuming that FE=($\mathsf{FE.Setup}$, $\mathsf{FE.Keygen}$,$\mathsf{FE.Enc}$, $\mathsf{FE.Dec}$) is a secure functional encryption scheme as per Definition 3, the outputs of experiments as described in $\mathcal{H}_{UAD,2}$ and $\mathcal{H}_{UAD,3}$ are computationally indistinguishable.

Proof. The only difference between the hybrids is the manner in which the challenge ciphertext is created. More specifically, in $\mathcal{H}_{UAD,2}$, the fourth component of the ciphertext is computed as an encryption of message x_0, whereas in $\mathcal{H}_{UAD,3}$ as an encryption of message x_1. Let us consider an adversary $\mathcal{A}_{\mathsf{FE}}$ that interacts with a challenger $\mathcal{C}$ to break the security of FE. Also, internally, it acts as the challenger in the PAFE security game against an adversary $\mathcal{A}_{\mathcal{H}_{UAD,2}-\mathcal{H}_{UAD,3}}$ that tries to distinguish between $\mathcal{H}_{UAD,2}$ and $\mathcal{H}_{UAD,3}$. $\mathcal{A}_{FE}$ functions as follows: it chooses to encrypt x_0, executes everything honestly for indices $\{1, 2, 3\}$, and forwards the pair (x_0,x_1) to challenger $\mathcal{C}$. $\mathcal{A}_{FE}$ receives a ciphertext ct_4, corresponding to either of the two plaintexts above and forwards it to $\mathcal{A}_{\mathcal{H}_{UAD,2}-\mathcal{H}_{UAD,3}}$ alongside the honestly generated information. $\mathcal{A}_{\mathcal{H}_{UAD,2}-\mathcal{H}_{UAD,3}}$ outputs a bit β which $\mathcal{A}_{FE}$ forwards to $\mathcal{C}$. Clearly, $\mathcal{A}_{FE}$ runs in polynomial time and $Adv^{sec-PAFE}(\mathcal{A}_{\mathcal{H}_{UAD,2}-\mathcal{H}_{UAD,3}}) \leq Adv^{FE-security}(\mathcal{A}_{\mathsf{FE}}) \leq \mathsf{negl}(\lambda)$. Therefore, $\mathcal{H}_{UAD,2}$ and $\mathcal{H}_{UAD,3}$ are computationally indistinguishable.

[$\mathcal{H}_{UAD,3} \approx \mathcal{H}_{UAD,4}$] Assuming that NIWI_f is a non-interactive witness indistinguishable proof system, the outputs of experiments as described in $\mathcal{H}_{UAD,3}$ and $\mathcal{H}_{UAD,4}$ are computationally indistinguishable.

Proof. The only difference between the two hybrids is the manner in which proof π_f is computed. In $\mathcal{H}_{UAD,3}$ is computed for condition (2) of $\mathsf{R}_{UAD,f}$ using the indices $\{1, 2, 3\}$, whereas in $\mathcal{H}_{UAD,4}$ is computed for condition (2) of $\mathsf{R}_{UAD,f}$ using indices $\{1, 2, 4\}$. Since NIWI_f is a non-interactive witness indistinguishable proof system, $\mathcal{H}_{UAD,3}$ and $\mathcal{H}_{UAD,4}$ are computationally indistinguishable.

[$\mathcal{H}_{UAD,4} \approx \mathcal{H}_{UAD,5}$] Assuming that FE=($\mathsf{FE.Setup}$, $\mathsf{FE.Keygen}$,$\mathsf{FE.Enc}$, $\mathsf{FE.Dec}$) is a secure functional encryption scheme as per Definition 3, the outputs of experiments as described in $\mathcal{H}_{UAD,4}$ and $\mathcal{H}_{UAD,5}$ are computationally indistinguishable.

Proof. The only difference between the hybrids is the manner in which the challenge ciphertext is created. More specifically, in $\mathcal{H}_{UAD,4}$, the third component of the ciphertext is computed as an encryption of message x_0, whereas in $\mathcal{H}_{UAD,5}$ as an encryption of message x_1. The proof is very similar with the one for $\mathcal{H}_{UAD,2} \approx \mathcal{H}_{UAD,3}$.

[$\mathcal{H}_{UAD,5} \approx \mathcal{H}_{UAD,6}$] Assuming that NIWI_d is a non-interactive witness indistinguishable proof system, the outputs of experiments as described in $\mathcal{H}_{UAD,5}$ and $\mathcal{H}_{UAD,6}$ are computationally indistinguishable.

Proof. The only difference between the two hybrids is the manner in which proofs π_d and π_f and computed. In $\mathcal{H}_{UAD,5}$ π_d is computed for condition (2) of $\mathsf{R}_{UAD,d}$ using the indices $\{1,2\}$, whereas in $\mathcal{H}_{UAD,6}$ is computed for condition (2) of $\mathsf{R}_{UAD,d}$ using the indices $\{1,4\}$. Since NIWI_d is a non-interactive witness indistinguishable proof systems, $\mathcal{H}_{UAD,5}$ and $\mathcal{H}_{UAD,6}$ are computationally indistinguishable.

[$\mathcal{H}_{UAD,6} \approx \mathcal{H}_{UAD,7}$] Assuming that NIWI_f is a non-interactive witness indistinguishable proof system, the outputs of experiments as described in $\mathcal{H}_{UAD,6}$ and $\mathcal{H}_{UAD,7}$ are computationally indistinguishable.

Proof. The only difference between the two hybrids is the manner in which proof π_f is computed. In $\mathcal{H}_{UAD,6}$ is computed for condition (2) of $\mathsf{R}_{UAD,f}$ using the indices $\{1,2,4\}$, whereas in $\mathcal{H}_{UAD,7}$ is computed for condition (2) of $\mathsf{R}_{UAD,f}$ using the indices $\{1,3,4\}$. Since NIWI_f is a non-interactive witness indistinguishable proof system, $\mathcal{H}_{UAD,6}$ and $\mathcal{H}_{UAD,7}$ are computationally indistinguishable.

[$\mathcal{H}_{UAD,7} \approx \mathcal{H}_{UAD,8}$] Assuming that FE=($\mathsf{FE.Setup}$,$\mathsf{FE.Keygen}$,$\mathsf{FE.Enc}$,$\mathsf{FE.Dec}$) is a secure functional encryption scheme as per Definition 3, the outputs of experiments as described in $\mathcal{H}_{UAD,7}$ and $\mathcal{H}_{UAD,8}$ are computationally indistinguishable.

Proof. The only difference between the hybrids is the manner in which the challenge ciphertext is created. More specifically, in $\mathcal{H}_{UAD,7}$, the second component of the ciphertext is computed as an encryption of message x_0, whereas in $\mathcal{H}_{UAD,8}$ as an encryption of message x_1. The proof is similar with the one for $\mathcal{H}_{UAD,2} \approx \mathcal{H}_{UAD,3}$.

[$\mathcal{H}_{UAD,8} \approx \mathcal{H}_{UAD,9}$] Assuming that NIWI_d is a non-interactive witness indistinguishable proof system, the outputs of experiments as described in $\mathcal{H}_{UAD,8}$ and $\mathcal{H}_{UAD,9}$ are computationally indistinguishable.

Proof. The only difference between the two hybrids is the manner in which proof π_d is computed. In $\mathcal{H}_{UAD,8}$ is computed for condition (2) of $\mathsf{R}_{UAD,d}$ using the trapdoor witness with indices $\{1,4\}$, whereas $\mathcal{H}_{UAD,9}$ is computed for condition (2) of $\mathsf{R}_{UAD,d}$ using the real witness for indices $\{3,4\}$. The proof is very similar to the one of $\mathcal{H}_{UAD,2} \approx \mathcal{H}_{UAD,3}$.

[$\mathcal{H}_{UAD,9} \approx \mathcal{H}_{UAD,10}$] Assuming that NIWI_f is a non-interactive witness indistinguishable proof system, the outputs of experiments as described in $\mathcal{H}_{UAD,9}$ and $\mathcal{H}_{UAD,10}$ are computationally indistinguishable.

Proof. The only difference between the two hybrids is the manner in which proof π_f is computed. In $\mathcal{H}_{UAD,9}$ is computed for condition (2) of $\mathsf{R}_{UAD,f}$ using the indices $\{1,3,4\}$, whereas in $\mathcal{H}_{UAD,10}$ is computed for condition (2) of $\mathsf{R}_{UAD,f}$ using the indices $\{2,3,4\}$. Since NIWI_f is a non-interactive witness indistinguishable proof system, $\mathcal{H}_{UAD,9}$ and $\mathcal{H}_{UAD,10}$ are computationally indistinguishable.

[$\mathcal{H}_{UAD,10} \approx \mathcal{H}_{UAD,11}$] Assuming that FE=(FE.Setup,FE.Keygen,FE.Enc, FE.Dec) is a secure functional encryption scheme as per Definition 3, the outputs of experiments as described in $\mathcal{H}_{UAD,10}$ and $\mathcal{H}_{UAD,11}$ are computationally indistinguishable.

Proof. The only difference between the hybrids is the manner in which the challenge ciphertext is created. More specifically, in $\mathcal{H}_{UAD,10}$, the first component of the ciphertext is computed as an encryption of message x_0, whereas in $\mathcal{H}_{UAD,11}$ as an encryption of message x_1. The proof is very similar with the one for $\mathcal{H}_{UAD,2} \approx \mathcal{H}_{UAD,3}$.

[$\mathcal{H}_{UAD,11} \approx \mathcal{H}_{UAD,12}$] Assuming that NIWI_f is a non-interactive witness indistinguishable proof system, the outputs of experiments as described in $\mathcal{H}_{UAD,11}$ and $\mathcal{H}_{UAD,12}$ are computationally indistinguishable.

Proof. The only difference between the two hybrids is the manner in which proof π_f is computed. In $\mathcal{H}_{UAD,11}$ is computed for condition (2) of $\mathsf{R}_{UAD,f}$ using the indices $\{2,3,4\}$, whereas in $\mathcal{H}_{UAD,12}$ is computed for condition (1) of $\mathsf{R}_{UAD,f}$ using the real witness with indices $\{1,2,3,4\}$.The proof is very similar to the one of $\mathcal{H}_{UAD,1} \approx \mathcal{H}_{UAD,2}$.

[$\mathcal{H}_{UAD,12} \approx \mathcal{H}_{UAD,13}$] Assuming that NIWI_d is a non-interactive witness indistinguishable proof system, the outputs of experiments as described in $\mathcal{H}_{UAD,12}$ and $\mathcal{H}_{UAD,13}$ are computationally indistinguishable.

Proof. The only difference between the two hybrids is the manner in which proof π_d is computed. In $\mathcal{H}_{UAD,12}$ is computed for condition (2) of $\mathsf{R}_{UAD,d}$ using the trapdoor witness with indices $\{3,4\}$, whereas $\mathcal{H}_{UAD,13}$ is computed for condition (1) of $\mathsf{R}_{UAD,d}$ using the real witness for indices $\{1,2,3,4\}$. The proof is very similar to the one of $\mathcal{H}_{UAD,0} \approx \mathcal{H}_{UAD,1}$.

(***Auditability***). For an adversary to break the auditability property for PAFE, it needs to provide the respective convincing proofs while the output of the decryption algorithm does not correspond to any function output. In addition to the analysis performed in the proofs of Theorem 1 and Theorem 2 we now need to consider all four possible combinations for accepting proofs π_d, π_f and show that in all possible cases the decryption output is the legitimate one.

1. Condition (1) of relation $R_{UAD,d}$ and condition (1) of relation $R_{UAD,f}$ are satisfied. Therefore, $\forall k \in [4]$: $\mathsf{pk}_{f,k} \leftarrow \mathsf{Com}(\mathsf{sk}_{f,k}; r_{f,k}) \wedge \mathsf{Dec}(\mathsf{mpk}_k, f, \mathsf{sk}_{f,k}, \mathsf{ct}_k) = \mathrm{y}_k$ and $\forall j \in [4]$: $\mathsf{pk}_{f,j} \leftarrow \mathsf{Com}(\mathsf{sk}_{f,j}; r_{f,j}) \wedge (\mathsf{mpk}_j, \mathsf{msk}_j) \leftarrow \mathsf{Setup}(1^\lambda; s_j) \wedge \mathsf{sk}_{f,j} \leftarrow \mathsf{KeyGen}(\mathsf{mpk}_j, \mathsf{msk}_j, f; r_{k,j})$. As $\forall k \in [4]$ $\mathsf{Dec}(\mathsf{mpk}_k, f_k, \mathsf{sk}_{f,k}, \mathsf{ct}_k) = \mathrm{y}_k$, therefore, $\mathsf{PAFE.Dec}(\{\mathsf{mpk}_i, f_i, \mathsf{sk}_{f,i}, \mathsf{ct}_i\}_{i \in [4]}) = \mathrm{y}_k$.
2. Condition (1) of relation $R_{UAD,d}$ and condition (2) of relation $R_{UAD,f}$ are satisfied. Therefore, $\forall k \in [4]$: $\mathsf{pk}_{f,k} \leftarrow \mathsf{Com}(\mathsf{sk}_{f,k}; r_{f,k}) \wedge \mathsf{Dec}(\mathsf{mpk}_k, f, \mathsf{sk}_{f,k}, \mathsf{ct}_k) = \mathrm{y}_k$ and $\exists A_1 \subset [4]$, with $|A_1| = 3$, s.t. $\forall j \in A_1$: $\mathsf{pk}_{f,j} \leftarrow \mathsf{Com}(\mathsf{sk}_{f,j}; r_{f,j}) \wedge (\mathsf{mpk}_j, \mathsf{msk}_j) \leftarrow \mathsf{Setup}(1^\lambda; s_j) \wedge \mathsf{sk}_{f,j} \leftarrow \mathsf{KeyGen}(\mathsf{mpk}_j, \mathsf{msk}_j, f; r_{k,j})$. Additionally, $\forall$ $i \in [4]$ $\mathsf{FE.Enc}(\mathsf{mpk}_i, \mathsf{ek}_i, \mathsf{x}) = \mathsf{ct}_i$. As $\exists A_1 \subset [4]$, with $|A_1| = 3$, s.t. $\mathsf{Dec}(\mathsf{mpk}_k, f_k, \mathsf{sk}_{f,k}, \mathsf{ct}_k) = \mathrm{y}_k$, $\mathsf{PAFE.Dec}(\{\mathsf{mpk}_i, f_i, \mathsf{sk}_{f,i}, \mathsf{ct}_i\}_{i \in A_1}) = \mathrm{y}_k$.
3. Condition (2) of relation $R_{UAD,d}$ and condition (1) of relation $R_{UAD,f}$ are satisfied. Therefore, $\exists A_2 \subset [4]$, with $|A_2| = 2$, such that $\forall k \in A_2$: $\mathsf{pk}_{f,k} \leftarrow \mathsf{Com}(\mathsf{sk}_{f,k}; r_{f,k}) \wedge \mathsf{Dec}(\mathsf{mpk}_k, f_k, \mathsf{sk}_{f,k}, \mathsf{ct}_k) = \mathsf{y}_k$ and $\forall j \in [4]$: $\mathsf{pk}_{f,j} \leftarrow \mathsf{Com}(\mathsf{sk}_{f,j}; r_{f,j}) \wedge (\mathsf{mpk}_j, \mathsf{msk}_j) \leftarrow \mathsf{Setup}(1^\lambda; s_j) \wedge \mathsf{sk}_{f,j} \leftarrow \mathsf{KeyGen}(\mathsf{mpk}_j, \mathsf{msk}_j, f_j; r_{k,j})$. Since the encryptor is trusted $\mathsf{ct} \leftarrow \mathsf{PAFE.Enc}(\mathsf{mpk}, \mathsf{ek}, \mathsf{x})$. Additionally all sk_f are validly generated. Therefore y_k is a valid decryption for all indexes.
4. Condition (2) of relation $R_{UAD,d}$ and condition (2) of relation $R_{UAD,f}$ are satisfied. Therefore, $\exists A_2 \subset [4]$, with $|A_2| = 2$, s.t. $\forall k \in A_2$: $\mathsf{pk}_{f,k} \leftarrow \mathsf{Com}(\mathsf{sk}_{f,k}; r_{f,k}) \wedge \mathsf{Dec}(\mathsf{mpk}_k, f_k, \mathsf{sk}_{f,k}, \mathsf{ct}_k) = \mathsf{y}_k \wedge \mathsf{c}_f = \mathsf{Com}(\{\mathsf{sk}_{f,i}\}_{i \in [4]}\}; u_f) \wedge \mathsf{c}_d = \mathsf{Com}(0; u_d)$ and $\exists A_1 \subset [4]$, with $|A_1| = 3$, s.t. $\forall j \in A_1$: $\mathsf{pk}_{f,j} \leftarrow \mathsf{Com}(\mathsf{sk}_{f,j}; r_{f,j}) \wedge (\mathsf{mpk}_j, \mathsf{msk}_j) \leftarrow \mathsf{Setup}(1^\lambda; s_j) \wedge \mathsf{sk}_{f,j} \leftarrow \mathsf{KeyGen}(\mathsf{mpk}_j, \mathsf{msk}_j, f_j; r_{k,j}) \wedge \mathsf{c}_f = \mathsf{Com}(\{\mathsf{sk}_{f,i}\}_{i \in [4]}\}; u_f) \wedge \exists\ \mathsf{y} \in \mathcal{X}_\lambda$ such that $\forall$ $i \in [4]$ $\mathsf{Dec}(\mathsf{mpk}_i, f_i, \mathsf{sk}_{f,i}, \mathsf{ct}_i) = \mathsf{y}$. By pigeonhole principle $\exists j^\star \in [4]$ for which $\mathsf{FE.Dec}(\mathsf{mpk}_{j^\star}, f_{j^\star}, \mathsf{sk}_{f,j^\star}, \mathsf{ct}_{j^\star}) = \mathrm{y}_k$. Additionally, similarly to the case above $\forall$ $i \in [4]$ $\mathsf{FE.Enc}(\mathsf{mpk}_i, \mathsf{ek}_i, \mathsf{x}) = \mathsf{ct}_i$. Thus, since 3-out-of-4 sk_f are validly generated as well, deterministically, $\mathsf{PAFE.Dec}(\{\mathsf{mpk}_i, f_i, \mathsf{sk}_{f,i}, \mathsf{ct}_i\}_{i \in A_1}) = \mathrm{y}_k$.

D Proof of Theorem 4

Proof (***Security***). To prove our constructions secure we show that the execution of the security game when the challenger chooses $b = 0$ is indistinguishable from the one when it picks $b = 1$. The proof comprises of a series of hybrids which are depicted in Table 2 and are consecutively indistinguishable. The proof has many similarities with that of Theorem 3, hence here we focus on some interesting steps where the two proofs deviate. Let us focus on the interesting cases of $\mathcal{H}_{UEAD,9} \approx \mathcal{H}_{UEAD,10}$ and the PA-UEAD auditability.

$\mathcal{H}$	$\{\mathsf{ct}_1,\mathsf{ct}_2,\mathsf{ct}_3,\mathsf{ct}_4\}$	π_e	c_e	$\{(\mathsf{sk}_f,\mathsf{pk}_f)_i\}_i$	π_f	c_f	y	π_d	c_d	Security
0	$\mathsf{x}_0,\mathsf{x}_0,\mathsf{x}_0,\mathsf{x}_0$	(1)	1	f,f,f,f	(1)	1	y_k	(1)	1	-
1	$\mathsf{x}_0,\mathsf{x}_0,\mathsf{x}_0,\mathsf{x}_0$	(1)	1	f,f,f,f	(1)	1	y_k	(1)	0	COM$_d$
2	$\mathsf{x}_0,\mathsf{x}_0,\mathsf{x}_0,\mathsf{x}'_0$	(1)	1	f,f,f,f	(1)	1	y_k	(2)	0	NIWI$_d$
3	$\mathsf{x}_0,\mathsf{x}_0,\mathsf{x}_0,\mathsf{x}'_0$	(1)	0	f,f,f,f	(1)	1	y_k	(2)	0	COM$_e$
4	$\mathsf{x}_0,\mathsf{x}_0,\mathsf{x}^\star_0,\mathsf{x}'^\star_0$	(2)	0	f,f,f,f	(1)	1	y_k	(2)	0	NIWI$_e$
5	$\mathsf{x}_0,\mathsf{x}_0,\mathsf{x}^\star_0,\mathsf{x}'^\star_0$	(2)	0	f,f,f,f	(1)	$\{\mathrm{sk}_{f,i}\}_i$	y_k	(2)	0	COM$_f$
6	$\mathsf{x}_0,\mathsf{x}_0,\mathsf{x}^\star_0,\mathsf{x}'^\star_0$	(2)	0	f,f,f,f'	(2)	$\{\mathrm{sk}_{f,i}\}_i$	y_k	(2)	0	NIWI$_f$
7	$\mathsf{x}_0,\mathsf{x}_0,\mathsf{x}^\star_0,\mathsf{x}'^\star_1$	(2)	0	f,f,f,f'	(2)	$\{\mathrm{sk}_{f,i}\}_i$	y_k	(2)	0	FE
8	$\mathsf{x}_0,\mathsf{x}_0,\mathsf{x}'^\star_0,\mathsf{x}^\star_1$	(2)	0	f,f,f',f	(2)	$\{\mathrm{sk}_{f,i}\}_i$	y_k	(2)	0	NIWI$_d$,NIWI$_f$
9	$\mathsf{x}_0,\mathsf{x}_0,\mathsf{x}'^\star_1,\mathsf{x}^\star_1$	(2)	0	f,f,f',f	(2)	$\{\mathrm{sk}_{f,i}\}_i$	y_k	(2)	0	FE
10	$\mathsf{x}^\star_0,\mathsf{x}'^\star_0,\mathsf{x}_1,\mathsf{x}_1$	(2)	0	f,f',f,f	(2)	$\{\mathrm{sk}_{f,i}\}_i$	y_k	(2)	0	NIWI$_d$,NIWI$_e$,NIWI$_f$
11	$\mathsf{x}^\star_0,\mathsf{x}'^\star_1,\mathsf{x}_1,\mathsf{x}_1$	(2)	0	f,f',f,f	(2)	$\{\mathrm{sk}_{f,i}\}_i$	y_k	(2)	0	FE
12	$\mathsf{x}'^\star_0,\mathsf{x}^\star_1,\mathsf{x}_1,\mathsf{x}_1$	(2)	0	f',f,f,f	(2)	$\{\mathrm{sk}_{f,i}\}_i$	y_k	(2)	0	NIWI$_d$,NIWI$_f$
13	$\mathsf{x}'^\star_1,\mathsf{x}^\star_1,\mathsf{x}_1,\mathsf{x}_1$	(2)	0	f',f,f,f	(2)	$\{\mathrm{sk}_{f,i}\}_i$	y_k	(2)	0	FE
14	$\mathsf{x}'^\star_1,\mathsf{x}^\star_1,\mathsf{x}_1,\mathsf{x}_1$	(2)	0	f,f,f,f	(1)	$\{\mathrm{sk}_{f,i}\}_i$	y_k	(2)	0	NIWI$_f$
15	$\mathsf{x}'^\star_1,\mathsf{x}^\star_1,\mathsf{x}_1,\mathsf{x}_1$	(2)	0	f,f,f,f	(1)	1	y_k	(2)	0	COM$_f$
16	$\mathsf{x}'_1,\mathsf{x}_1,\mathsf{x}_1,\mathsf{x}_1$	(1)	0	f,f,f,f	(1)	1	y_k	(2)	0	NIWI$_e$
17	$\mathsf{x}'_1,\mathsf{x}_1,\mathsf{x}_1,\mathsf{x}_1$	(1)	1	f,f,f,f	(1)	1	y_k	(2)	0	COM$_e$
18	$\mathsf{x}_1,\mathsf{x}_1,\mathsf{x}_1,\mathsf{x}_1$	(1)	1	f,f,f,f	(1)	1	y_k	(1)	0	NIWI$_d$
19	$\mathsf{x}_1,\mathsf{x}_1,\mathsf{x}_1,\mathsf{x}_1$	(1)	1	f,f,f,f	(1)	1	y_k	(1)	1	COM$_d$

Table 3. Hybrids for the security proof of the PA-UEAD PAFE.

- Hybrid $\mathcal{H}_{UEAD,0}$: This is the security game with challenge bit $b = 0$. The master public key is $\mathsf{mpk}' = \{\mathsf{mpk}_i\}_{i\in[4]}$, where $\forall i \in [4]$ $(\mathsf{msk}_i,\mathsf{mpk}_i,\mathsf{ek}_i) \leftarrow \mathsf{Setup}(1^\lambda;s_i)$ for random string s_i. $\mathsf{c}_f \leftarrow \mathsf{Com}(1^{4\cdot len};u_f)$, $\mathsf{c}_e \leftarrow \mathsf{Com}(1^{len};u_e)$, and $\mathsf{c}_d \leftarrow \mathsf{Com}(1^{len};u_d)$ for random strings u_f,u_e,u_d. The challenge ciphertext is $\mathsf{ct}' \leftarrow \{\mathsf{ct}_i\}_{i\in[4]}$, where $\forall i \in [4]$ $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{ek}_i,\mathsf{x}_0;\mathsf{r}_{e,i})$, for some random string $\mathsf{r}_{e,i}$. The functional secret keys are $\mathsf{sk}'_f = \{\mathsf{sk}_{f,i},r_{f,i}\}_{i\in[4]}$, where $\forall i \in [4]$ $\mathsf{sk}_{f,i} \leftarrow \mathsf{FE.KeyGen}(\mathsf{mpk}_i,\mathsf{msk}_i,\ f_i;r_{k_i})$, for random strings $\mathsf{r}_{f,i}$ and $\mathsf{r}_{k,i}$. The functional public keys are $\mathsf{pk}_f = \{\mathsf{pk}_{f,i}\}_{i\in[4]}$, where $\forall i \in [4]$ $\mathsf{pk}_{f,i} \leftarrow \mathsf{Com.Commit}(\mathsf{sk}_{f,i};r_{f,i})$. The function output $\mathsf{y}_0 \leftarrow \mathsf{FE.Dec}(\mathsf{mpk}_i,f,\mathsf{sk}_{f,i},\mathsf{ct}_i)$, $\forall i \in [4]$. π_f is computed for condition (1) of relation $\mathsf{R}_{UAD,f}$ and π_d is computed for condition (1) of relation $\mathsf{R}_{UAD,d}$.
- Hybrid $\mathcal{H}_{UEAD,1}$: This hybrid is identical to the previous one except for c_e, which is computed differently, i.e., $\mathsf{c}_d \leftarrow \mathsf{Com.Commit}(0;u_d)$.
- Hybrid $\mathcal{H}_{UEAD,2}$: This hybrid is identical to the previous one, except for the computation of π_d. Using indices $\{1,2\}$, π_d is computed for condition (2) of relation $\mathsf{R}_{UEAD,d}$.
- Hybrid $\mathcal{H}_{UEAD,3}$: They hybrid is identical to the previous one except for c_e, which is computed differently, i.e., $\mathsf{c}_f \leftarrow \mathsf{Com.Commit}(0;u_e)$.
- Hybrid $\mathcal{H}_{UEAD,4}$: This hybrid is identical to the previous one, except for the computation of π_e. Using indices $\{1,2\}$, π_e is computed for condition (2) of relation $\mathsf{R}_{UEAD,e}$.

- Hybrid $\mathcal{H}_{UEAD,5}$: This hybrid is identical to the previous one except for c_f, which is computed differently, i.e., $\mathsf{c}_f \leftarrow \mathsf{Com.Commit}(0^{4\cdot len}; u_f)$.
- Hybrid $\mathcal{H}_{UEAD,6}$: This hybrid is identical to the previous one, except for the computation of π_f. Using indices $\{1,2,3\}$, π_f is computed for condition (2) of relation $\mathsf{R}_{UEAD,f}$.
- Hybrid $\mathcal{H}_{UEAD,7}$: This hybrid is identical to the previous one, except for the computation of ct_4, i.e., $\mathsf{ct}_4 \leftarrow \mathsf{FE.Enc}(\mathsf{ek}_4{,}\mathsf{x}_1{;}\mathsf{r}_{e,4})$.
- Hybrid $\mathcal{H}_{UEAD,8}$: This hybrid is identical to the previous one, except for the computation of π_f, which is computed using indices $\{1,2,4\}$.
- Hybrid $\mathcal{H}_{UEAD,9}$: This hybrid is identical to the previous one, except for ct_3, i.e., $\mathsf{ct}_3 \leftarrow \mathsf{FE.Enc}(\mathsf{ek}_3{,}\mathsf{x}_1{;}\mathsf{r}_{e,3})$.
- Hybrid $\mathcal{H}_{UEAD,10}$: This hybrid is identical to the previous one, except for the computation of π_d and π_e which are computed using indices $\{3,4\}$, and π_f which is computed using indices $\{1,3,4\}$.
- Hybrid $\mathcal{H}_{UEAD,11}$: This hybrid is identical to the previous one, except for the computation of ct_2, i.e., $\mathsf{ct}_2 \leftarrow \mathsf{Fe.Enc}(\mathsf{ek}_2{,}\mathsf{x}_1{;}\mathsf{r}_{e,2})$.
- Hybrid $\mathcal{H}_{UEAD,12}$: This hybrid is identical to the previous one, except for the computation of π_f, which is computed using indices $\{2,3,4\}$.
- Hybrid $\mathcal{H}_{UEAD,13}$: This hybrid is identical to the previous one, except for ct_1, i.e., $\mathsf{ct}_1 \leftarrow \mathsf{FE.Enc}(\mathsf{ek}_1{,}\mathsf{x}_1{;}\mathsf{r}_{e,1})$.
- Hybrid $\mathcal{H}_{UEAD,14}$: This hybrid is identical to the previous one, except for the computation of π_f, which is computed for condition (1) using indices $\{1,2,3,4\}$.
- Hybrid $\mathcal{H}_{UEAD,15}$: This hybrid is identical to the previous one except for c_d, which is computed differently, i.e., $\mathsf{c}_d \leftarrow \mathsf{Com.Commit}(1; u_f)$.
- Hybrid $\mathcal{H}_{UEAD,16}$: This hybrid is identical to the previous one, except for the computation of π_e. Using indices $\{1,2,3,4\}$, π_e is computed for condition (1) of relation $\mathsf{R}_{UEAD,e}$.
- Hybrid $\mathcal{H}_{UEAD,17}$: This hybrid is identical to the previous one except for c_e, which is computed differently, i.e., $\mathsf{c}_e \leftarrow \mathsf{Com.Commit}(1; u_e)$.
- Hybrid $\mathcal{H}_{UEAD,18}$: This hybrid is identical to the previous one, except for the computation of π_d. Using indices $\{1,2,3,4\}$, π_d is computed for condition (1) of relation $\mathsf{R}_{UEAD,d}$.
- Hybrid $\mathcal{H}_{UEAD,19}$: This hybrid is identical to the previous one except for c_d, which is computed differently, i.e., $\mathsf{c}_d \leftarrow \mathsf{Com.Commit}(1; u_d)$, which is identical to the real game when $b = 1$.

$[\mathcal{H}_{UEAD,0} \approx \mathcal{H}_{UEAD,1}]$ Assuming that Com is a computationally hiding commitment scheme, the outputs of experiments as described in $\mathcal{H}_{UEAD,0}$ and $\mathcal{H}_{UEAD,1}$ are computationally indistinguishable.

Proof. The only difference between the two hybrids is the manner in which the commitment c_d is computed. Let us consider an adversary $\mathcal{A}_{\mathsf{Com}}$ that interacts with a challenger $\mathcal{C}$ to break the hiding property of the commitment scheme. Also, internally, it acts as the challenger in the PAFE security game against an adversary $\mathcal{A}_{UEAD,\mathcal{H}_0-\mathcal{H}_1}$ that tries to distinguish between $\mathcal{H}_{UEAD,0}$ and $\mathcal{H}_{UEAD,1}$. $\mathcal{A}_{Com}$ functions as follows: it chooses to encrypt x_0, executes everything honestly, and forwards two stings, namely (0^{len}) and (1^{len}) to challenger $\mathcal{C}$. $\mathcal{A}_{Com}$ receives a commitment c_d, corresponding to either of the two strings above and forwards it to $\mathcal{A}_{UEAD,\mathcal{H}_0-\mathcal{H}_1}$ alongside the honestly generated information. $\mathcal{A}_{UEAD,\mathcal{H}_0-\mathcal{H}_1}$ outputs a bit β which $\mathcal{A}_{Com}$ forwards to $\mathcal{C}$. Clearly, $\mathcal{A}_{Com}$ runs in polynomial time and $Adv^{sec-PAFE}(\mathcal{A}) \leq Adv^{Com-Hiding}(\mathcal{A}_{Com}) \leq \mathsf{negl}(\lambda)$. Therefore, $\mathcal{H}_{UEAD,0}$ and $\mathcal{H}_{UEAD,1}$ are computationally indistinguishable.

$[\mathcal{H}_{UEAD,1} \approx \mathcal{H}_{UEAD,2}]$ Assuming that NIWI_d is a non-interactive witness indistinguishable proof system, the outputs of experiments as described in $\mathcal{H}_{UEAD,1}$ and $\mathcal{H}_{UEAD,2}$ are computationally indistinguishable.

Proof. The only difference between the two hybrids is the manner in which proof π_d is computed. In $\mathcal{H}_{UEAD,1}$ is computed for condition (1) of $\mathsf{R}_{UEAD,d}$ using the "real" witness for indices $\{1, 2, 3, 4\}$, whereas in $\mathcal{H}_{UEAD,2}$ is computed for condition (2) of $\mathsf{R}_{UEAD,d}$ using the "trapdoor" witness for indices $\{1, 2\}$. Similarly to the previous lemma we can construct an adversary that plays the witness indistinguishability game for NIWI_d, and internally acts as the challenger for the indistinguishability game of PAFE. The advantage of the NIWI-adversary is bound by the PAFE-adversary, and since NIWI_d is a non-interactive witness indistinguishable proof system, $\mathcal{H}_{UEAD,1}$ and $\mathcal{H}_{UEAD,2}$ are computationally indistinguishable.

$[\mathcal{H}_{UEAD,2} \approx \mathcal{H}_{UEAD,3}]$ Assuming that Com is a computationally hiding commitment scheme, the outputs of experiments as described in $\mathcal{H}_{UEAD,2}$ and $\mathcal{H}_{UEAD,3}$ are computationally indistinguishable.

Proof. The only difference between the two hybrids is the manner in which the commitment c_e is computed. The proof is similar to the one of $[\mathcal{H}_{UEAD,0} \approx \mathcal{H}_{UEAD,1}]$.

$[\mathcal{H}_{UEAD,3} \approx \mathcal{H}_{UEAD,4}]$ Assuming that NIWI_e is a non-interactive witness indistinguishable proof system, the outputs of experiments as described in $\mathcal{H}_{UEAD,3}$ and $\mathcal{H}_{UEAD,4}$ are computationally indistinguishable.

Proof. The only difference between the two hybrids is the manner in which proof π_e is computed. In $\mathcal{H}_{UEAD,3}$ is computed for condition (1) of $\mathsf{R}_{UEAD,e}$ using the real witness using indices $\{1, 2, 3, 4\}$, whereas in $\mathcal{H}_{UEAD,4}$ is computed for condition (2) of $\mathsf{R}_{UEAD,e}$ using the indices $\{1, 2\}$. The proof is very similar to the one of $[\mathcal{H}_{UEAD,1} \approx \mathcal{H}_{UEAD,2}]$.

$[\mathcal{H}_{UEAD,4} \approx \mathcal{H}_{UEAD,5}]$ Assuming that Com is a computationally hiding commitment scheme, the outputs of experiments as described in $\mathcal{H}_{UEAD,4}$ and $\mathcal{H}_{UEAD,5}$ are computationally indistinguishable.

Proof. The only difference between the two hybrids is the manner in which the commitment c_f is computed. The proof is similar to the one of $[\mathcal{H}_{UEAD,0} \approx \mathcal{H}_{UEAD,1}]$.

[$\mathcal{H}_{UEAD,5} \approx \mathcal{H}_{UEAD,6}$] Assuming that NIWI_f is a non-interactive witness indistinguishable proof system, the outputs of experiments as described in $\mathcal{H}_{UEAD,5}$ and $\mathcal{H}_{UEAD,6}$ are computationally indistinguishable.

Proof. The only difference between the two hybrids is the manner in which proof π_e is computed. In $\mathcal{H}_{UEAD,5}$ is computed for condition (1) of $\mathsf{R}_{UEAD,f}$ using the real witness using indices $\{1,2,3,4\}$, whereas in $\mathcal{H}_{UEAD,6}$ is computed for condition (2) of $\mathsf{R}_{UEAD,f}$ using the indices $\{1,2,3\}$. The proof is very similar to the one of [$\mathcal{H}_{UEAD,1} \approx \mathcal{H}_{UEAD,2}$].

[$\mathcal{H}_{UEAD,6} \approx \mathcal{H}_{UEAD,7}$] Assuming that FE is a secure functional encryption scheme as per Definition 3, the outputs of experiments as described in $\mathcal{H}_{UEAD,6}$ and $\mathcal{H}_{UEAD,7}$ are computationally indistinguishable.

Proof. The only difference between the hybrids is the manner in which the challenge ciphertext is created. More specifically, in $\mathcal{H}_{UEAD,6}$, the fourth component of the ciphertext is computed as an encryption of message x_0, whereas in $\mathcal{H}_{UEAD,7}$ as an encryption of message x_1. Let us consider an adversary $\mathcal{A}_{\mathsf{FE}}$ that interacts with a challenger $\mathcal{C}$ to break the security of FE. Also, internally, it acts as the challenger in the PAFE security game against an adversary $\mathcal{A}_{UEAD,H_6-H_7}$ that tries to distinguish between $\mathcal{H}_{UEAD,6}$ and $\mathcal{H}_{UAED,7}$. $\mathcal{A}_{FE}$ functions as follows: it chooses to encrypt x_0, executes everything honestly for indices $\{1,2,3\}$, and forwards the pair $(\mathsf{x}_0,\mathsf{x}_1)$ to challenger $\mathcal{C}$. $\mathcal{A}_{FE}$ receives a ciphertext ct_4, corresponding to either of the two plaintexts above and forwards it to $\mathcal{A}_{UEAD,H_6-H_7}$ alongside the honestly generated information. $\mathcal{A}_{UAD,H_6-H_7}$ outputs a bit β which $\mathcal{A}_{FE}$ forwards to $\mathcal{C}$. Clearly, $\mathcal{A}_{FE}$ runs in polynomial time and $Adv^{sec-PAFE}(\mathcal{A}) \leq Adv^{FE-security}(\mathcal{A}_{FE}) \leq \mathsf{negl}(\lambda)$. Therefore, $\mathcal{H}_{UEAD,6}$ and $\mathcal{H}_{UEAD,7}$ are computationally indistinguishable.

[$H_7 \approx H_8$] Assuming that NIWI_f is a non-interactive witness indistinguishable proof system, the outputs of experiments as described in $\mathcal{H}_{UEAD,7}$ and $\mathcal{H}_{UEAD,8}$ are computationally indistinguishable.

Proof. The only difference between the two hybrids is the manner in which proof π_f is computed. In $\mathcal{H}_{UEAD,7}$ is computed for condition (2) of $\mathsf{R}_{UEAD,f}$ using the indices $\{1,2,3\}$, whereas in $\mathcal{H}_{UEAD,8}$ is computed for condition (2) of $\mathsf{R}_{UEAD,f}$ using the indices $\{1,2,4\}$. Since NIWI_f is a non-interactive witness indistinguishable proof system, $\mathcal{H}_{UEAD,7}$ and $\mathcal{H}_{UAED,8}$ are computationally indistinguishable.

[$\mathcal{H}_{UEAD,8} \approx \mathcal{H}_{UEAD,9}$] Assuming that FE is a secure functional encryption scheme as per Definition 3, the outputs of experiments as described in $\mathcal{H}_{UEAD,8}$ and $\mathcal{H}_{UEAD,9}$ are computationally indistinguishable.

Proof. The only difference between the hybrids is the manner in which the challenge ciphertext is created. More specifically, in $\mathcal{H}_{UAED,8}$, the third component of the ciphertext is computed as an encryption of message x_0, whereas in $\mathcal{H}_{UEAD,9}$ as an encryption of message x_1. The proof is very similar with the one for [$\mathcal{H}_{UEAD,4} \approx \mathcal{H}_{UEAD,5}$].

[$\mathcal{H}_{UEAD,9} \approx \mathcal{H}_{UEAD,10}$] Assuming that NIWI_d, NIWI_e, NIWI_f are non-interactive witness indistinguishable proof systems, the outputs of experiments as described in $\mathcal{H}_{UEAD,9}$ and $\mathcal{H}_{UEAD,10}$ are computationally indistinguishable.

Proof. The only difference between the two hybrids is the manner in which proofs π_d, π_e, and π_f and computed. In $\mathcal{H}_{UEAD,9}$ π_d is computed for condition (2) of $\mathsf{R}_{UEAD,d}$ using indices $\{1,2,4\}$, whereas in $\mathcal{H}_{UEAD,10}$ is computed for condition (2) of $\mathsf{R}_{UEAD,d}$ using indices $\{1,3,4\}$. Similarly, π_e is computed for condition (2) of $\mathsf{R}_{UEAD,e}$ using the indices $\{1,2\}$, whereas in $\mathcal{H}_{UEAD,10}$ is computed for condition (2) of $\mathsf{R}_{UEAD,e}$ using indices $\{3,4\}$. In $\mathcal{H}_{UEAD,9}$ π_f is computed for condition (2) of $\mathsf{R}_{UEAD,f}$ using indices $\{1,2,4\}$, whereas in $\mathcal{H}_{UEAD,10}$ is computed for condition (2) of $\mathsf{R}_{UEAD,f}$ using the indices $\{1,3,4\}$. Observe that the two hybrids are symmetrical to each other. By doing these three changes at once we can guarantee the existence and knowledge of the respective witnesses. Furthermore, we can argue that if there exists a PPT adversary that can distinguish between the two hybrids, then we can construct an adversary $\mathcal{A}_{\mathsf{NIWI}}$ that can break at least one of NIWI$_f$, NIWI$_e$, or NIWI$_d$ witness indistinguishability property.

[$\mathcal{H}_{UEAD,10} \approx \mathcal{H}_{UEAD,11}$] Assuming that FE is a secure functional encryption scheme as per Definition 3, the outputs of experiments as described in $\mathcal{H}_{UEAD,10}$ and $\mathcal{H}_{UEAD,11}$ are computationally indistinguishable.

Proof. The only difference between the hybrids is the manner in which the challenge ciphertext is created. More specifically, in $\mathcal{H}_{UEAD,10}$, the second component of the ciphertext is computed as an encryption of message x_0, whereas in $\mathcal{H}_{UEAD,11}$ as an encryption of message x_1. The proof is similar with the one for [$\mathcal{H}_{UEAD,4} \approx \mathcal{H}_{UEAD,5}$].

[$\mathcal{H}_{UEAD,11} \approx \mathcal{H}_{UEAD,12}$] Assuming that NIWI_f is a non-interactive witness indistinguishable proof system, the outputs of experiments as described in $\mathcal{H}_{UEAD,11}$ and $\mathcal{H}_{UEAD,12}$ are computationally indistinguishable.

Proof. The only difference between the two hybrids is the manner in which proof π_f is computed. In $\mathcal{H}_{UEAD,11}$ is computed for condition (2) of $\mathsf{R}_{UEAD,f}$ using the indices $\{1,2,4\}$, whereas in $\mathcal{H}_{UEAD,12}$ is computed for condition (2) of $\mathsf{R}_{UEAD,f}$ using the indices $\{2,3,4\}$. Since NIWI_f is a non-interactive witness indistinguishable proof system, $\mathcal{H}_{UEAD-11}$ and $\mathcal{H}_{UEAD,12}$ are computationally indistinguishable.

[$\mathcal{H}_{UEAD,12} \approx \mathcal{H}_{UEAD,13}$] Assuming that FE is a secure functional encryption scheme as per Definition 3, the outputs of experiments as described in $\mathcal{H}_{UEAD,12}$ and $\mathcal{H}_{UEAD,13}$ are computationally indistinguishable.

Proof. The only difference between the hybrids is the manner in which the challenge ciphertext is created. More specifically, in $\mathcal{H}_{UEAD,12}$, the first component of the ciphertext is computed as an encryption of message x_0, whereas in $\mathcal{H}_{UEAD,13}$ as an encryption of message x_1. The proof is very similar with the one for [$\mathcal{H}_{UEAD,4} \approx \mathcal{H}_{UEAD,5}$].

$[\mathcal{H}_{UEAD,13} \approx \mathcal{H}_{UEAD,14}]$ Assuming that NIWI_f is a non-interactive witness indistinguishable proof system, the outputs of experiments as described in $\mathcal{H}_{UEAD,13}$ and $\mathcal{H}_{UEAD,14}$ are computationally indistinguishable.

Proof. The only difference between the two hybrids is the manner in which proof π_f is computed. In $\mathcal{H}_{UEAD,13}$ is computed for condition (2) of $\mathsf{R}_{UEAD,f}$ using the indices $\{2, 3, 4\}$, whereas in $\mathcal{H}_{UEAD,14}$ is computed for condition (1) of $\mathsf{R}_{UAD,f}$ using the real witness for indices $\{1, 2, 3, 4\}$. Since NIWI_f is a non-interactive witness indistinguishable proof system, $\mathcal{H}_{UEAD,13}$ and $\mathcal{H}_{UEAD,14}$ are computationally indistinguishable.

$[\mathcal{H}_{UEAD,14} \approx \mathcal{H}_{UEAD,15}]$ Assuming that Com is a computationally hiding commitment scheme, the outputs of experiments as described in $\mathcal{H}_{UEAD,14}$ and $\mathcal{H}_{UEAD,15}$ are computationally indistinguishable.

Proof. The only difference between the two hybrids is the manner in which the commitment c_f is computed. The proof is similar to the one of $[\mathcal{H}_{UEAD,0} \approx \mathcal{H}_{UEAD,1}]$.

$[\mathcal{H}_{UEAD,15} \approx \mathcal{H}_{UEAD,16}]$ Assuming that NIWI_e is a non-interactive witness indistinguishable proof system, the outputs of experiments as described in $\mathcal{H}_{UEAD,15}$ and $\mathcal{H}_{UEAD,16}$ are computationally indistinguishable.

Proof. The only difference between the two hybrids is the manner in which proof π_e is computed. In $\mathcal{H}_{UEAD,15}$ is computed for condition (2) of $\mathsf{R}_{UEAD,e}$ using the trapdoor witness with indices $\{3, 4\}$, whereas $\mathcal{H}_{UEAD,16}$ is computed for condition (1) of $\mathsf{R}_{UEAD,d}$ using the real witness for indices $\{1, 2, 3, 4\}$. The proof is very similar to the one of $[\mathcal{H}_{UEAD,1} \approx \mathcal{H}_{UEAD,2}]$.

$[\mathcal{H}_{UEAD,16} \approx \mathcal{H}_{UEAD,17}]$ Assuming that Com is a computationally hiding commitment scheme, the outputs of experiments as described in $\mathcal{H}_{UEAD,16}$ and $\mathcal{H}_{UEAD,17}$ are computationally indistinguishable.

Proof. The only difference between the two hybrids is the manner in which the commitment c_e is computed. The proof is similar to the one of $[\mathcal{H}_{UEAD,0} \approx \mathcal{H}_{UEAD,1}]$.

$[\mathcal{H}_{UEAD,17} \approx \mathcal{H}_{UEAD,18}]$ Assuming that NIWI_d is a non-interactive witness indistinguishable proof system, the outputs of experiments as described in $\mathcal{H}_{UEAD,17}$ and $\mathcal{H}_{UEAD,18}$ are computationally indistinguishable.

Proof. The only difference between the two hybrids is the manner in which proof π_d is computed. In $\mathcal{H}_{UEAD,17}$ is computed for condition (2) of $\mathsf{R}_{UEAD,d}$ using the indices $\{3, 4\}$, whereas in $\mathcal{H}_{UEAD,18}$ is computed for condition (1) of $\mathsf{R}_{UEAD,d}$ using the real witness with indices $\{1, 2, 3, 4\}$.The proof is very similar to the one of $[\mathcal{H}_{UEAD,1} \approx \mathcal{H}_{UEAD,2}]$.

$[\mathcal{H}_{UEAD,18} \approx \mathcal{H}_{UEAD,19}]$ Assuming that Com is a computationally hiding commitment scheme, the outputs of experiments as described in $\mathcal{H}_{UEAD,18}$ and $\mathcal{H}_{UEAD,19}$ are computationally indistinguishable.

Proof. The only difference between the two hybrids is the manner in which the commitment c_d is computed. The proof is similar to the one of $[\mathcal{H}_{UEAD,0} \approx \mathcal{H}_{UEAD,1}]$.

(***Auditability***). For an adversary to break the auditability property for PAFE, it needs to provide the respective convincing proofs while the output of the decryption algorithm is not the respective function output. We now consider all eight possible combinations for accepting proofs π_f, π_e, π_d and show that in all possible cases the provided function output is the legitimate one.

1. Condition (1) of relation $R_{UEAD,f}$, condition (1) of relation $R_{UEAD,e}$, and condition (1) of relation $R_{UEAD,d}$ are satisfied. Now, for all four instances $(\mathsf{mpk}_j,\mathsf{msk}_j) \leftarrow \mathsf{FE.Setup}(1^\lambda; s_j)$, $\mathsf{sk}_{f,j} \leftarrow \mathsf{FE.KeyGen}(\mathsf{mpk}_j,\mathsf{msk}_j,f_j;r_{f,j})$, $\mathsf{pk}_{f,j} \leftarrow \mathsf{Com.Commit}(\mathsf{sk}_{f,j}; r_{f,j})$, $\mathsf{ct}_j = \mathsf{FE.Enc}(\{\mathsf{ek}_j,\mathsf{mpk}_j\},\mathsf{x};r_{e,j})$, and moreover $\mathsf{FE.Dec}(\mathsf{mpk}_k,\ f_k,\mathsf{sk}_{f,k},\ \mathsf{ct}_k)=\mathrm{y}_k$, Informally, as all keys are generated honestly, x is encrypted in all $\{\mathsf{ct}\}_{i\in[4]}$ and all decryptions output y. From the public auditability definition, PA-UEAD holds.
2. Condition (1) of relation $R_{UEAD,f}$, condition (1) of relation $R_{UEAD,e}$, and condition (2) of relation $R_{UEAD,d}$ are satisfied. In a similar manner to case 1 and the PA-UAD one, since $\forall j \in [4] : \mathsf{ct}_j = \mathsf{FE.Enc}(\{\mathsf{ek}_j,\mathsf{mpk}_j\},\mathsf{x};r_{e,j})$, $(\mathsf{mpk}_j,\mathsf{msk}_j) \leftarrow \mathsf{FE.Setup}(1^\lambda; s_j)$, $\mathsf{pk}_{f,j} \leftarrow \mathsf{Com.Commit}(\mathsf{sk}_{f,j}; r_{f,j})$, $\mathsf{sk}_{f,j} \leftarrow \mathsf{FE.KeyGen}(\mathsf{mpk}_j,\mathsf{msk}_j,f_j;r_{f,j})$ it therefore must be that the decryption output $\mathsf{PAFE.Dec}(\{\mathsf{mpk}_i,f_i,\mathsf{sk}_{f,i},\mathsf{ct}_i\}_{i\in[4]})=\mathrm{y}_k$. From π_d we get that at least three decryptions output y_f,which combined with x being encrypted to all ciphertexts guarantees PA-UEAD.
3. Condition (1) of relation $R_{UEAD,f}$, condition (2) of relation $R_{UEAD,e}$, and condition (1) of relation $R_{UEAD,d}$ are satisfied. This is an impossible case as π_e proves that $\mathsf{c}_e \leftarrow \mathsf{Com}(0^{len}; u_e)$ and π_d contrary proves $\mathsf{c}_e \leftarrow \mathsf{Com}(1^{len}; u_e)$.
4. Condition (1) of relation $R_{UEAD,f}$, condition (2) of relation $R_{UEAD,e}$, and condition (2) of relation $R_{UEAD,d}$ are satisfied. This concretely states that $\forall j : (\mathsf{mpk}_j,\mathsf{msk}_j) \leftarrow \mathsf{FE.Setup}(1^\lambda; s_j)$, $\mathsf{sk}_{f,j} \leftarrow \mathsf{FE.KeyGen}(\mathsf{mpk}_j,\mathsf{msk}_j,f_j;r_{f,j})$, $\mathsf{pk}_{f,j} \leftarrow \mathsf{Com.Commit}(\mathsf{sk}_{f,j}; r_{f,j})$, for two $\mathsf{ct}_j = \mathsf{FE.Enc}(\{\mathsf{ek}_j,\mathsf{mpk}_j\},\mathsf{x};r_{e,j})$, and for three $\mathsf{FE.Dec}(\mathsf{mpk}_k,\ f_k,\mathsf{sk}_{f,k},\ \mathsf{ct}_k)=\mathrm{y}_k$. Informally, in this case all keys, two ciphertexts and three decryptions are generated honestly. By the pigeonhole principle at least one decryption will use a ciphertext whose index was used as a witness in the generation of π_e. This, combined with the fact that three decryptions (all using honestly created keys) return the same function output guarantees the PA-UEAD property.
5. Condition (2) of relation $R_{UEAD,f}$, condition (1) of relation $R_{UEAD,e}$, and condition (1) of relation $R_{UEAD,d}$ are satisfied. This is an impossible case as π_f proves that $\mathsf{c}_f \leftarrow \mathsf{Com.Commit}(\{\mathsf{sk}_i\}_{i\in[4]}; u_f)$ and π_e contrary proves $\mathsf{c}_f \leftarrow \mathsf{Com.Commit}(1^{len}; u_e)$.
6. Condition (2) of relation $R_{UEAD,f}$, condition (1) of relation $R_{UEAD,e}$, and condition (2) of relation $R_{UEAD,d}$ are satisfied. This is an impossible case since on one hand π_f proves that $\mathsf{c}_f \leftarrow \mathsf{Com}(\{\mathsf{sk}_i\}_{i\in[4]}; u_f)$ and π_e on the other proves $\mathsf{c}_f \leftarrow \mathsf{Com}(1^{len}; u_e)$.

7. Condition (2) of relation $R_{UEAD,f}$, condition (2) of relation $R_{UEAD,e}$, and condition (1) of relation $R_{UEAD,d}$ are satisfied. This is an impossible case as π_e proves that $\mathsf{c}_e \leftarrow \mathsf{Com}(0^{len}; u_e)$ and π_d contrary proves $\mathsf{c}_e \leftarrow \mathsf{Com}(1^{len}; u_e)$.
8. Condition (2) of relation $R_{UEAD,f}$, condition (2) of relation $R_{UEAD,e}$, and condition (2) of relation $R_{UEAD,d}$ are satisfied. In this case π_f ensures that $\exists$ $\mathsf{y} \in \mathcal{X}_\lambda$ such that $\forall$ $i \in [4]$: $\mathsf{y} \leftarrow \mathsf{FE.Dec}(\mathsf{mpk}_i, f_i, \mathsf{sk}_{f,i}, \mathsf{ct}_i)$. This combined with the pigeonhole principle about the ciphertexts and the decryptions (similarly to case 4), satisfies the PA-UEAD property.

Importantly, in all cases above where PA-UEAD is achieved, it is not restricted to a single function. Recall that public auditability states that the decryption returns a function output, implying consistency of the encryptor's input. In PA-UAD it is trivial since the encryptor is trusted and always encrypts x. In PA-UEAD though, this derives from the combination of either satisfied condition for $\mathsf{R}_{UEAD,d}$ and either a combination of $\mathsf{R}_{UEAD,f}$ condition (1) with any of the $\mathsf{R}_{UEAD,e}$ conditions, or a combination of $\mathsf{R}_{UEAD,f}$ condition (2) with $\mathsf{R}_{UEAD,e}$ condition (2). Specifically, for every function there exist at least three instances that return the same function output for f and from which at least one of them corresponds to the function evaluation of an encryptor-chosen plaintext x.

E Proof of Theorem 5

Proof. **(Security)** We first consider the cases where the adversary follows a strategy specified in cases (*) or (**) in Theorem 1. The general case analysis applies to our MIFE construction and thus no PPT adversary $\mathcal{A}_{\mathsf{PAFE}}$ cannot have more than $\frac{1}{2}$ probability of winning its game, in these cases.

Case (i): Now we prove $\mathcal{M}$ secure by contraposition for the remaining case. We consider a PPT adversary $\mathcal{A}_{\mathsf{PAFE}}$ that has non-negligible advantage ϵ of winning the IND-Security game for $\mathcal{M}$. We will construct an adversary $\mathcal{A}_{\mathsf{MIFE}}$ that utilizes $\mathcal{A}_{\mathsf{PAFE}}$ and has also non-negligible advantage in winning the IND-security game for $\mathcal{M}'$.

– Initialization: $\mathcal{A}_{\mathsf{MIFE}}$ receives mpk from the challenger $\mathcal{C}$, samples $j^\star \leftarrow\!\!\$\ [m]$, initializes $counter = 0$, initializes a table $\mathcal{T}_{enc}$, samples $r_s \leftarrow \{0,1\}^\lambda$, computes $\mathsf{c}_d = g^\top \cdot h^{u_d}$, samples $b' \leftarrow \{0,1\}$, and forwards $(\mathsf{mpk}, \mathsf{c}_d)$ to $\mathcal{A}_{\mathsf{PAFE}}$.
– Encryption queries: When $\mathcal{A}_{\mathsf{PAFE}}$ issues a QEnc(x_0,x_1) query to $\mathcal{A}_{\mathsf{MIFE}}$, the latter issues a QEnc(x_j,x_j) query to $\mathcal{C}$ and increments $counter$ by 1. $x_j = x_0$ for $counter < j^\star$, and $x_j = x_1$ for $counter > j^\star$. For $counter = j^\star$ $\mathcal{A}_{\mathsf{MIFE}}$ forwards the query to $\mathcal{C}$ without any alteration. $\mathcal{C}$ returns a ciphertext ct, which $\mathcal{A}_{\mathsf{MIFE}}$ forwards to $\mathcal{A}_{\mathsf{PAFE}}$.
– Functional secret key queries: When $\mathcal{A}_{\mathsf{PAFE}}$ issues a QSKeyGen query to $\mathcal{A}_{\mathsf{MIFE}}$, the latter forwards the query to $\mathcal{C}$, who responds with $\mathsf{sk}_f = \big((\Sigma_{i=1}^n s_i \cdot w_i, \Sigma_{i=1}^n t_i \cdot w_i)\big)$. $\mathcal{A}$ then checks if a QPKeyGen query has been issued for f. If not, it

samples $r_f \leftarrow\!\!\$ \{0,1\}^\lambda$ and computes $\mathsf{pk}_f{=}(g^{\Sigma_{i=1}^n s_i \cdot w_i} \cdot h^{r_f}, h^{\Sigma_{i=1}^n t_i \cdot w_i} \cdot g^{r_f})$. $\mathcal{A}_{\mathsf{FE}}$ forwards $(\mathsf{sk}_f,\mathsf{pk}_f)$ to $\mathcal{A}_{\mathsf{PAFE}}$.

- Functional public key queries: When $\mathcal{A}_{\mathsf{PAFE}}$ issues a QPKeyGen(f) query to $\mathcal{A}_{\mathsf{MIFE}}$, the latter checks whether $\exists \mathsf{ct}$ associated with a QEnc(x_0, x_1) such that $f(\mathsf{x}_0) \neq f(\mathsf{x}_1)$. If so, $\mathcal{A}_{\mathsf{MIFE}}$ samples $z_f \leftarrow\!\!\$ \{0,1\}^\lambda$ and computes $\mathsf{pk}_f{=}(g^{z_f}, h^{z_f})$. Otherwise, $\mathcal{A}_{\mathsf{MIFE}}$ forwards a QSKeyGen(f) query to $\mathcal{C}$, who responds with $\mathsf{sk}_f{=}(\Sigma_{i=1}^n s_i \cdot w_i, \Sigma_{i=1}^n t_i \cdot w_i)$. $\mathcal{A}_{\mathsf{MIFE}}$ samples $r_f \leftarrow\!\!\$ \{0,1\}^\lambda$, and computes $pk_f = (g^{\Sigma_{i=1}^n s_i \cdot w_i} \cdot h^{r_f}, h^{\Sigma_{i=1}^n t_i \cdot w_i} \cdot g^{r_f})$. In any case $\mathcal{A}_{\mathsf{MIFE}}$ returns pk_f to $\mathcal{A}_{\mathsf{PAFE}}$.
- Decryption queries: When $\mathcal{A}_{\mathsf{PAFE}}$ issues a QDec(ct, f) query to $\mathcal{A}_{\mathsf{MIFE}}$, the latter assigns $\mathsf{y} \leftarrow f(\mathsf{x}_j)$ and $\pi_d \leftarrow \mathsf{NIWI}_d.\mathsf{Prove}(\mathsf{mpk}, \top, \bot, u_d, \bot, \bot, \bot, \mathsf{c}_d)$. $\mathcal{A}_{\mathsf{MIFE}}$ forwards (y, π_d) to $\mathcal{A}_{\mathsf{PAFE}}$.
- Finalization: $\mathcal{A}_{\mathsf{PAFE}}$ outputs a bit b' which $\mathcal{A}_{\mathsf{MIFE}}$ forwards to $\mathcal{C}$.

$\mathcal{A}_{\mathsf{MIFE}}$ runs in polynomial-time as it runs the probabilistic polynomial-time (PPT) adversary $\mathcal{A}_{\mathsf{PAFE}}$ and additionally performs polynomial-time operations (i.e. exponentiations and multiplications). The differences between the view provided by $\mathcal{A}_{\mathsf{MIFE}}$ in the reduction and the view provided by a challenger while executing the real security game for PAFE lie in the computation of the functional public keys, the ciphertexts, the function outputs, the commitment c_d, and the proofs π_d. Our goal is to prove that $Adv^{sec-MIFE}(\mathcal{A}_{\mathsf{MIFE}}(1^\lambda)) \leq Adv^{sec-PAFE}(\mathcal{A}_{\mathsf{PAFE}}(1^\lambda)) + negl(\lambda)$.

We can use the hybrid analysis of Theorem 1. From the employed MIFE and NIZK, we just need to argue about the indistinguishability of the hybrids involving the changes in the commitments c_d and pk_f. The former follows the same distribution as in the game execution so the two hybrids are indistinguishable. The same holds for the functional public keys. Last, remember that the choice of $j^\star$ induces a polynomial loss in the advantage of $\mathcal{A}_{\mathsf{MIFE}}$.

Additionally, we consider the strategy of $\mathcal{A}_{\mathsf{PAFE}}$ that corresponds to case *(ii)* in Theorem 1. Assuming the existance of a PPT adversary $\mathcal{A}_{\mathsf{PAFE,(ii)}}$ that wins in its game with more than negligible advantage ϵ, we construct a PPT adversary $\mathcal{A}_{\mathsf{MIFE,(ii)}}$ that utilizes $\mathcal{A}_{\mathsf{PAFE,(ii)}}$ to win in its game with also non-negligible advantage.

- Initialization: $\mathcal{A}_{\mathsf{MIFE,(ii)}}$ receives $mpk = (\mathbb{G}, g, h, \{h_i\}_{i \in [n]})$ from $\mathcal{C}$, samples $r_d \leftarrow\!\!\$ \{0,1\}^\lambda$, computes $c_d = g^\top \cdot h^{r_d}$ and runs $\mathcal{A}_{\mathsf{PAFE,(ii)}}$ on input $(1^\lambda, mpk, c_d)$.
- Encryption queries: When $\mathcal{A}_{\mathsf{PAFE,(ii)}}$ issues an encryption query, $\mathcal{A}_{\mathsf{MIFE,(ii)}}$ forwards it to $\mathcal{C}$, receives the ciphertext ct_i and gives it to $\mathcal{A}_{\mathsf{PAFE,(ii)}}$.
- Functional secret key queries: When $\mathcal{A}_{\mathsf{PAFE},(ii)}$ issues a QSKeyGen query to $\mathcal{A}_{\mathsf{MIFE},(ii)}$, the latter forwards the query to $\mathcal{C}$, who responds with sk_f. $\mathcal{A}_{\mathsf{FE}}$ then checks if a QPKeyGen query has been issued for f. If not, it samples $r_f \leftarrow\!\!\$ \{0,1\}^\lambda$ and computes $\mathsf{pk}_f \leftarrow \mathsf{Com.Commit}(\mathsf{sk}_f; r_f)$, $\mathcal{A}_{\mathsf{MIFFE}_{(ii)}}$ forwards $(\mathsf{sk}_f,\mathsf{pk}_f)$ to $\mathcal{A}_{\mathsf{PAFE}_{(ii)}}$.
- Functional public key queries: When $\mathcal{A}_{\mathsf{PAFE,(ii)}}$ issues a query, $\mathcal{A}_{\mathsf{MIFE,(ii)}}$ samples $r_f \leftarrow\!\!\$ \{0,1\}^\lambda$ and computes $\mathsf{pk}_f{=}\ (g^{r_f}, h^{r_f})$ and forwards it to $\mathcal{A}_{\mathsf{PAFE,(ii)}}$.

– Decryption queries: When $\mathcal{A}_{\mathsf{PAFE}}$ issues a QDec(ct, f) query to $\mathcal{A}_{\mathsf{FE}}$, the latter assigns $\mathsf{y} \leftarrow f(\mathsf{x}_j)$ and $\pi_d \leftarrow \mathsf{NIZK}_d.\mathsf{Prove}(\mathsf{mpk},\{w_i\}_{i\in[n]},\perp,\mathsf{pk}_f,\mathsf{ct},\mathsf{y})$. $\mathcal{A}_{\mathsf{FE}}$ forwards (y,π_d) to $\mathcal{A}_{\mathsf{PAFE}}$.
– Finalization: $\mathcal{A}_{\mathsf{PAFE,(ii)}}$ sends b' to $\mathcal{A}_{\mathsf{MIFE,(ii)}}$ who forwards it to $\mathcal{C}$.

In this case we argue that the view of $\mathcal{A}_{\mathsf{PAFE,(ii)}}$ is indistinguishable from the real game. The only difference lies in the computation of the commitment c_d and the functional public keys. First, we state that c_d follows the same distribution in both cases as it is randomly sampled. Therefore, we argue that the indistinguishability of the two views reduces to the indistinguishability of the functional public keys. Similarly to c_d they follow the same distribution, as they are also randomly sampled in both cases. From this analysis we conclude that $Adv^{sec-MIFE}(\mathcal{A}_{\mathsf{MIFE,(ii)}}) = Adv^{sec-PAFE}(\mathcal{A}_{\mathsf{PAFE,(ii)}})$, which is non-negligible.

(Auditability) Similarly to Theorem 1 for an adversary to break the auditability property they need to violate either the NIZK or the commitment condition. From the soundness property of the employed NIZK no PPT adversary can generate a convincing proof without knowing the witnesses and since *(i)* the symbol "$\top$" is not in the domain of the msk and *(ii)* the domain of the msk is superpolynomially big, the probability of a PPT adversary guessing it is negligible. Therefore, no PPT adversary can break our PA-UD MIFE auditability property.

References

1. Delloite-US. https://www2.deloitte.com/us/en.html
2. Facebook Community Standards. https://www.facebook.com/communitystandards
3. KPMG-CN. https://home.kpmg/cn/en/home.html
4. Abdalla, M., Benhamouda, F., Gay, R.: From single-input to multi-client inner-product functional encryption. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology - ASIACRYPT 2019. Lecture Notes in Computer Science, vol. 11923, pp. 552–582. Springer (2019). https://doi.org/10.1007/978-3-030-34618-8_19, https://doi.org/10.1007/978-3-030-34618-8_19
5. Agrawal, S., Goyal, R., Tomida, J.: Multi-input quadratic functional encryption from pairings. In: Malkin, T., Peikert, C. (eds.) Advances in Cryptology - CRYPTO 2021. Lecture Notes in Computer Science, vol. 12828, pp. 208–238. Springer (2021). https://doi.org/10.1007/978-3-030-84259-8_8, https://doi.org/10.1007/978-3-030-84259-8_8
6. Agrawal, S., Goyal, R., Tomida, J.: Multi-party functional encryption. In: Nissim, K., Waters, B. (eds.) TCC 2021. Lecture Notes in Computer Science, vol. 13043, pp. 224–255. Springer (2021). https://doi.org/10.1007/978-3-030-90453-1_8, https://doi.org/10.1007/978-3-030-90453-1_8
7. Agrawal, S., Libert, B., Stehlé, D.: Fully secure functional encryption for inner products, from standard assumptions. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology - CRYPTO 2016. Lecture Notes in Computer Science, vol. 9816, pp. 333–362. Springer (2016). https://doi.org/10.1007/978-3-662-53015-3_12, https://doi.org/10.1007/978-3-662-53015-3_12
8. Ambrona, M., Fiore, D., Soriente, C.: Controlled functional encryption revisited: Multi-authority extensions and efficient schemes for quadratic functions. Proc.

Priv. Enhancing Technol. **2021**(1), 21–42 (2021). https://doi.org/10.2478/popets-2021-0003, https://doi.org/10.2478/popets-2021-0003

9. Badertscher, C., Kiayias, A., Kohlweiss, M., Waldner, H.: Consistency for functional encryption. In: 34th IEEE Computer Security Foundations Symposium, CSF 2021, Dubrovnik, Croatia, June 21-25, 2021. pp. 1–16. IEEE (2021). https://doi.org/10.1109/CSF51468.2021.00045, https://doi.org/10.1109/CSF51468.2021.00045
10. Badrinarayanan, S., Goyal, V., Jain, A., Sahai, A.: Verifiable functional encryption. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology - ASIACRYPT 2016. Lecture Notes in Computer Science, vol. 10032, pp. 557–587 (2016). https://doi.org/10.1007/978-3-662-53890-6_19, https://doi.org/10.1007/978-3-662-53890-6_19
11. Baltico, C.E.Z., Catalano, D., Fiore, D., Gay, R.: Practical functional encryption for quadratic functions with applications to predicate encryption. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology - CRYPTO 2017. Lecture Notes in Computer Science, vol. 10401, pp. 67–98. Springer (2017). https://doi.org/10.1007/978-3-319-63688-7_3, https://doi.org/10.1007/978-3-319-63688-7_3
12. Barbosa, M., Farshim, P.: Delegatable homomorphic encryption with applications to secure outsourcing of computation. In: Dunkelman, O. (ed.) Topics in Cryptology - CT-RSA 2012 - The Cryptographers' Track at the RSA Conference 2012, San Francisco, CA, USA, February 27 - March 2, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7178, pp. 296–312. Springer (2012). https://doi.org/10.1007/978-3-642-27954-6_19, https://doi.org/10.1007/978-3-642-27954-6_19
13. Bellare, M., Palacio, A.: GQ and schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In: Yung, M. (ed.) Advances in Cryptology - CRYPTO 2002. Lecture Notes in Computer Science, vol. 2442, pp. 162–177. Springer (2002). https://doi.org/10.1007/3-540-45708-9_11, https://doi.org/10.1007/3-540-45708-9_11
14. Bitansky, N., Paneth, O.: Zaps and non-interactive witness indistinguishability from indistinguishability obfuscation. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. Lecture Notes in Computer Science, vol. 9015, pp. 401–427. Springer (2015). https://doi.org/10.1007/978-3-662-46497-7_16, https://doi.org/10.1007/978-3-662-46497-7_16
15. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications (extended abstract). In: Simon, J. (ed.) Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA. pp. 103–112. ACM (1988). https://doi.org/10.1145/62212.62222, https://doi.org/10.1145/62212.62222
16. Boneh, D., Sahai, A., Waters, B.: Functional encryption: Definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. Lecture Notes in Computer Science, vol. 6597, pp. 253–273. Springer (2011). https://doi.org/10.1007/978-3-642-19571-6_16, https://doi.org/10.1007/978-3-642-19571-6_16
17. Camenisch, J., Stadler, M.: Efficient group signature schemes for large groups (extended abstract). In: Jr., B.S.K. (ed.) Advances in Cryptology - CRYPTO '97. Lecture Notes in Computer Science, vol. 1294, pp. 410–424. Springer (1997). https://doi.org/10.1007/BFb0052252, https://doi.org/10.1007/BFb0052252
18. Chatzigiannis, P., Baldimtsi, F.: Miniledger: Compact-sized anonymous and auditable distributed payments. In: Bertino, E., Shulman, H., Waidner, M. (eds.)

Computer Security - ESORICS 2021. Lecture Notes in Computer Science, vol. 12972, pp. 407–429. Springer (2021). https://doi.org/10.1007/978-3-030-88418-5_20, https://doi.org/10.1007/978-3-030-88418-5_20

19. Chotard, J., Dufour-Sans, E., Gay, R., Phan, D.H., Pointcheval, D.: Dynamic decentralized functional encryption. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology - CRYPTO 2020. Lecture Notes in Computer Science, vol. 12170, pp. 747–775. Springer (2020). https://doi.org/10.1007/978-3-030-56784-2_25, https://doi.org/10.1007/978-3-030-56784-2_25
20. Chotard, J., Sans, E.D., Gay, R., Phan, D.H., Pointcheval, D.: Decentralized multi-client functional encryption for inner product. In: Peyrin, T., Galbraith, S.D. (eds.) Advances in Cryptology - ASIACRYPT 2018. Lecture Notes in Computer Science, vol. 11273, pp. 703–732. Springer (2018). https://doi.org/10.1007/978-3-030-03329-3_24, https://doi.org/10.1007/978-3-030-03329-3_24
21. Confessore, N.: Cambridge analytica and facebook: The scandal and the fallout so far. https://www.nytimes.com/2018/04/04/us/politics/cambridge-analytica-scandal-fallout.html (2018)
22. Gentry, C.: A fully homomorphic encryption scheme. Ph.D. thesis, Stanford University, USA (2009), https://searchworks.stanford.edu/view/8493082
23. Goldwasser, S., Gordon, S.D., Goyal, V., Jain, A., Katz, J., Liu, F., Sahai, A., Shi, E., Zhou, H.: Multi-input functional encryption. In: Nguyen, P.Q., Oswald, E. (eds.) Advances in Cryptology - EUROCRYPT 2014. Lecture Notes in Computer Science, vol. 8441, pp. 578–602. Springer (2014). https://doi.org/10.1007/978-3-642-55220-5_32, https://doi.org/10.1007/978-3-642-55220-5_32
24. Gong, J., Qian, H.: Simple and efficient FE for quadratic functions. Des. Codes Cryptogr. **89**(8), 1757–1786 (2021). https://doi.org/10.1007/s10623-021-00871-x, https://doi.org/10.1007/s10623-021-00871-x
25. Goyal, V., Jain, A., O'Neill, A.: Multi-input functional encryption with unbounded-message security. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology - ASIACRYPT 2016. Lecture Notes in Computer Science, vol. 10032, pp. 531–556 (2016). https://doi.org/10.1007/978-3-662-53890-6_18, https://doi.org/10.1007/978-3-662-53890-6_18
26. Kang, H., Dai, T., Jean-Louis, N., Tao, S., Gu, X.: Fabzk: Supporting privacy-preserving, auditable smart contracts in hyperledger fabric. In: DSN 2019. pp. 543–555. IEEE (2019). https://doi.org/10.1109/DSN.2019.00061, https://doi.org/10.1109/DSN.2019.00061
27. Koutsos, V., Papadopoulos, D., Chatzopoulos, D., Tarkoma, S., Hui, P.: Agora: A privacy-aware data marketplace. IEEE Trans. Dependable Secur. Comput. **19**(6), 3728–3740 (2022). https://doi.org/10.1109/TDSC.2021.3105099, https://doi.org/10.1109/TDSC.2021.3105099
28. Libert, B., Titiu, R.: Multi-client functional encryption for linear functions in the standard model from LWE. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology - ASIACRYPT 2019. Lecture Notes in Computer Science, vol. 11923, pp. 520–551. Springer (2019). https://doi.org/10.1007/978-3-030-34618-8_18, https://doi.org/10.1007/978-3-030-34618-8_18
29. Narula, N., Vasquez, W., Virza, M.: zkledger: Privacy-preserving auditing for distributed ledgers. In: Banerjee, S., Seshan, S. (eds.) NSDI 2018. pp. 65–80. USENIX Association (2018), https://www.usenix.org/conference/nsdi18/presentation/narula
30. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) Advances in Cryptology - EUROCRYPT 2005. Lecture Notes in Computer Science,

vol. 3494, pp. 457–473. Springer (2005). https://doi.org/10.1007/11426639_27, https://doi.org/10.1007/11426639_27

31. Schoenmakers, B.: Cryptographic protocols. Lecture Notes, Department of Mathematics and Computer Science, Technical University of Eindhoven (2019)
32. Shafagh, H., Burkhalter, L., Hithnawi, A., Duquennoy, S.: Towards Blockchain-based Auditable Storage and Sharing of IoT Data. In: ACM CCSW@CCS 2017. pp. 45–50
33. Soroush, N., Iovino, V., Rial, A., Rønne, P.B., Ryan, P.Y.A.: Verifiable inner product encryption scheme. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020. Lecture Notes in Computer Science, vol. 12110, pp. 65–94. Springer (2020). https://doi.org/10.1007/978-3-030-45374-9_3, https://doi.org/10.1007/978-3-030-45374-9_3
34. Suzuki, T., Emura, K., Ohigashi, T., Omote, K.: Verifiable functional encryption using intel SGX. In: Huang, Q., Yu, Y. (eds.) ProvSec 2021. Lecture Notes in Computer Science, vol. 13059, pp. 215–240. Springer (2021). https://doi.org/10.1007/978-3-030-90402-9_12, https://doi.org/10.1007/978-3-030-90402-9_12
35. Tomida, J.: Tightly secure inner product functional encryption: Multi-input and function-hiding constructions. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology - ASIACRYPT 2019. Lecture Notes in Computer Science, vol. 11923, pp. 459–488. Springer (2019). https://doi.org/10.1007/978-3-030-34618-8_16, https://doi.org/10.1007/978-3-030-34618-8_16