(Inner-Product) Functional Encryption with Updatable Ciphertexts

Valerio Cini[1], Sebastian Ramacher[1], Daniel Slamanig[1], Christoph Striecks[1], and Erkan Tairi[2]

[1] AIT Austrian Institute of Technology, Vienna, Austria
{firstname.lastname}@ait.ac.at
[2] TU Wien, Vienna, Austria
erkan.tairi@tuwien.ac.at

Abstract. We propose a novel variant of functional encryption which supports ciphertext updates, dubbed ciphertext updatable functional encryption (CUFE). Such a feature further broadens the practical applicability of the functional encryption paradigm and is carried out via so-called update tokens. However, allowing update tokens requires some care for the security definition as we want that updates can be done by any semi-trusted third party and only on ciphertexts. Our contribution is three-fold:

a) We define our new primitive with a security notion in the indistinguishability setting. Within CUFE, functional decryption keys *and* ciphertexts are labeled with tags such that only if the tag of the decryption key and the ciphertext match, then decryption succeeds. Furthermore, we allow ciphertexts to switch their tags to any other tag via update tokens. Such tokens are generated by the holder of the main secret key and can only be used in the desired direction.
b) We present a generic construction of CUFE for any functionality as well as predicates different from equality testing on tags, which relies on the existence of (probabilistic) indistinguishability obfuscation (iO).
c) We present a practical construction of CUFE for the inner-product functionality from standard assumptions (i.e., LWE) in the random-oracle model. On the technical level, we build on the recent functional encryption schemes with fine-grained access control and linear operations on encrypted data (Abdalla et al., AC'20) and introduce an additional ciphertext updatability feature. Proving security for such a construction turned out to be non-trivial, particularly when revealing keys for the updated challenge ciphertext is allowed. Overall, such construction enriches the set of known inner-product functional-encryption schemes with the additional updatability feature of ciphertexts.

1 Introduction

Functional encryption [SW05, BSW11, O'N10] is an exciting encryption paradigm that allows fine-grained access control over encrypted data. In contrast to conventional encryption, which is all-or-nothing, in functional encryption (FE) there is a main secret key msk that allows to generate constrained functional decryption keys. More precisely, every decryption key sk_f is associated to a function f and given an encryption $\mathsf{Enc}(mpk, x)$ of some message x under the main public key mpk, the decryption with sk_f only reveals $f(x)$, but nothing more about x.[3]

Since its introduction, FE has been subject to intense study which can broadly be categorized into two areas. Firstly, works that consider general functionalities and thereby mostly focusing on feasibility results. This typically results in constructions beyond practical interest, as they rely on indistinguishability obfuscation (iO) or need to impose severe restrictions on the number of keys given to an adversary. Secondly, works that restrict the power by only supporting limited classes of functions that are of particular interest for practical applications, i.e., linear and quadratic functions. Here, the main focus is then on concrete and efficient constructions. One such approach that attracted a lot of research are FE schemes for the inner-product functionality (IPFE), i.e., keys are associated to vectors $\vec{y}$, messages are vectors $\vec{x}$ and decryption reveals $\langle \vec{x}, \vec{y} \rangle$. Initially proposed by Abdalla et al. [ABDP15], a line of work improved the security guarantees [ALS16, ABDP16, BBL17, CLT18], extended it to the multi-input [AGRW17, ACF+18] as well as the decentralized setting [CDG+18, ABKW19, LT19, ABG19, ABM+20]. Although this functionality

[3] Unless mentioned otherwise, we will always assume public-key functional encryption.

is very simple, it has already shown to be useful in privacy-preserving machine learning [MSH+19], data analytics,[4] or data marketplaces [KPC+20].

Limitations of large-scale deployment of FE. A problem for the practical adoption of FE is that every issued functional decryption key inherently leaks some information. For the inner-product functionality and thus IPFE this is particularly problematic. Specifically, if n is the dimension of the vectors, then obtaining n decryption keys in general allows to recover the full plaintext. Consequently, as soon as IPFE is deployed in some larger-scale setting, this represents a severe limitation. To mitigate this problem and make IPFE more practical, Abdalla, Catalano, Gay, and Ursu [ACGU20] recently introduced the notion of IPFE with fine-grained access control providing strong security guarantees.[5] Loosely speaking, the idea is that ciphertexts are produced with respect to an access policy (e.g., expressed by monotone span programs) and decryption keys are in addition to being bound to a function also associated to an attribute. Decryption then only works if the attribute in the key satisfies the access-policy in the ciphertext. It is important to stress that when aiming for reasonable security which allows collusion of functional decryption keys, this approach is non-trivial as a naive composition of IPFE with attribute-based encryption (ABE) or identity-based encryption (IBE) suffers from simple mix-and-match attacks. Abdalla et al. provide pairing-based attribute-based constructions covering monotone span programs (AB-IPFE) and lattice-based identity-based constructions (IB-IPFE). Nguyen et al. [NPP22] propose more efficient pairing-based constructions and investigate the approach of Abdalla et al. in a multi-client setting. Recently, Lai et al. [LLW21] as well as Pal and Dutta [PD21] also present lattice-based AB-IPFE constructions.

This concept of Abdalla et al. firstly mitigates the leakage problem of plain IPFE, as now this inherent limitation on the number of issued functional decryption key only applies per identity in IB-IPFE (or attribute-policy in AB-IPFE). This can be viewed as partitioning the keys such that the aforementioned limitation applies to each of these partitions, making it much more scalable. Secondly, it more closely reflects the situation in large-scale systems where even in the case of FE, one wants to enforce a more fine-grained control over who is allowed to learn some particular information of the encrypted plaintexts. Thirdly, this concept overcomes the problem of a trivial approach, i.e., encrypting data separately under an IPFE public key for each recipient, which would result in a linear blow-up of the ciphertexts.

Motivation towards more flexibility in fine-grained access control. Abdalla et al. [ACGU20] make an important step towards applicability of FE in large-scale systems. But it still seems limited when it comes to dynamic aspects. For instance, the medical example used in [ACGU20] envisions that doctors in a hospital may be able to compute on a different set of encrypted data than employees of a health insurance company. What happens if the access to data for the insurance company should be expanded? This would either mean to encrypt *all* the data anew under the policy that is satisfied by the insurance company or to issue additional keys to the insurance company. While in this medical setting this might still be manageable, there are other examples where this seems hard to achieve.

Let us therefore consider the emerging domain of data marketplaces.[6] These are platforms that allow customers to buy access to data or statistical analysis on data offered by a potentially huge set of data owners via data brokers. The available data sets can range from business intelligence and research, demographic or health, firmographic, and market data to public data. (IP)FE seems to be an interesting tool for this application. But while the use of IPFE (in a multi-client setting) has recently been proposed in [KPC+20] to realize a privacy-aware data marketplace, it does so in a way that it reveals the evaluations in plain to the data brokers. Now one could imagine using the approach in [ACGU20] to let data owners encrypt their data under certain policies (or identities), whereas data buyers are given functional keys (with respect to a certain identity or attribute) and data brokers basically only distribute the data (and possibly perform some aggregation tasks).

[4] https://research.kudelskisecurity.com/2021/02/02/benchmarking-privacy-preserving-motion-detection/

[5] There is more related work such as [DP19, AJS18, JLMS19, JLS19, CZY19, Wee17] as discussed in [ACGU20], but those schemes either provide less functionality or weaker security.

[6] https://research.aimultiple.com/data-marketplace, https://datarade.ai/platform-categories/personal-data-marketplaces

Still, it seems cumbersome to have a fine-grained control over what buyers can access if the access policies are fixed in the ciphertexts.

We now envision that in addition to having a fine-grained control, we allow the data brokers to update the policies (attributes/identities) in existing ciphertexts in order to add more flexibility. Let us now focus on the specific case of policies being represented via the equality predicate, and thus ciphertexts and function keys are labeled and decryption yields the function of the message if both labels match. We call those labels *tags*.[7] Data brokers should have the capability to update ciphertexts in a way that they can change the tags in ciphertexts using some additional information (called an update token), but they should not learn the function evaluations and thus the privacy of the data of the owners is guaranteed. To keep a fine-grained control over ciphertext updates in such a broker scenario, we want to restrict the updates of a ciphertext to a single update and the token to only work in one direction, i.e., from tag t to t' but not vice versa. Thus, already updated ciphertexts cannot be updated anymore. While it is possible to consider schemes that support multiple updates and/or bidirectional tokens, we believe that this is rather dangerous in such applications. For instance, this could allow moving ciphertexts to tags for which they were not intended, e.g., from a tag t to t' and then to t'' via two updates, whereas it might be not intended that it is possible to move all ciphertexts from t to t'', but rather only ones under t to t' and ones under t' to t''.

We note that this functionality goes beyond what is provided by IPFE with fine-grained access control due to Abdalla et al. [ACGU20] (but as we will see it can serve as a starting point). And a trivial construction based upon [ACGU20] that encrypts a message multiple times under different tags (identities) in parallel fails to provide the desired functionality. In particular, it does not allow to dynamically decide to which tag a ciphertext can be updated as the desired tags would have to be known at the time of producing the ciphertext, something that we want to avoid in our approach to solve the above problem! Consequently, we are looking for a solution where we can potentially switch a ciphertext to any tag from a large (i.e., exponential) tag space.

Since currently (IP)FE schemes that achieve the desired properties are absent in the cryptographic literature, in this work we ask:

Can we define and construct (IP)FE schemes with fine-grained access control and ciphertext updatability?

1.1 Our Contribution

We answer the above question affirmative via our three-fold contribution:

a) We define a new primitive dubbed ciphertext updatable functional encryption (CUFE) with security notion in the indistinguishability setting. Within CUFE, functional decryption keys *and* ciphertexts are labeled with tags such that only if the tag in the decryption key and ciphertext match, then decryption succeeds. Furthermore, we allow fresh ciphertexts to update its tags to any other tag via so-called update tokens and by any semi-trusted third party. Such tokens are generated by the key holder of the main secret and can only be used in the desired direction.
b) We present a generic construction of CUFE for any functionality and more powerful predicates than equality testing on tags, which relies on the existence of (probabilistic) indistinguishability obfuscation (iO).
c) We present a practical construction of CUFE for the inner-product functionality from standard assumptions (i.e., the learning-with-errors (LWE) assumption) in the random-oracle model. Proving security for such a construction turned out to be non-trivial, particularly when revealing keys for the updated challenge ciphertext is allowed. In general, this further enriches the approach presented in line of Abdalla et al. [ACGU20] with the updatability feature of ciphertexts. Notably, our construction relies on lattice-based assumptions which are plausibly post-quantum.

Defining ciphertext updatability for FE. CUFE can be seen as tag-based FE scheme with tag space $\mathcal{T}$. As in FE, key generation outputs a main public-secret key pair (mpk, msk), where the decryption keys $sk_{f,t}$ for some function $f \in \mathcal{F}$ and tag $t \in \mathcal{T}$ are derived from msk. In CUFE,

[7] One can also think of these labels as identities.

however, msk is also used to derive update tokens $\Delta_{t\to t'}$. Now, encryption takes some tag t and message x and outputs a ciphertext C_t. Then, using $\Delta_{t\to t'}$, any semi-trusted third party can update C_t to $C_{t'}$. Correctness guarantees that if the tags of the function key and the ciphertext match, and only a single update has happened, then decryption succeeds and outputs $f(x)$.

Defining security needs some care as we want that tokens can update ciphertexts only towards the tag specified in the update token and updated ciphertext should not be allowed to be further updated. That is, a token $\Delta_{t\to t'}$ can only switch tags from t to t' and not vice versa. As in the work of Abdalla et al. [ACGU20], the adversary is allowed to query decryption keys for any functionality f such that the function evaluation on the challenge ciphertext yields $f(x_0) = f(x_1)$, for adversarially chosen messages x_0, x_1, if the policy is fulfilled. In our constructions, we restrict the policy to the equality test on tags of the functional decryption key and the ciphertext (we discuss extensions in Section 4.2) which ensures a simple access control for our envisioned applications.

Concerning updated ciphertext, we have the following situation. Since the concept of update tokens is not foreseen in conventional forms of FE, we need to consider additional aspects for our security notions. We have to deal with the fact that tokens can potentially not only be used to update ciphertexts from some tag t to another tag t', but could also be used to invert a ciphertext update. This is partly reminiscent of providing adequate and strong security guarantees in proxy re-encryption (PRE) [Coh19, DKL+18]. Having those in mind, we define an indistinguishability-based notion IND-CUFE-CPA, which guarantees that an adversary cannot distinguish ciphertexts for a certain challenge target tag and adversarially chosen messages.

More concretely, as outlined in our motivation, we only want to allow updating the tags of ciphertexts *once* and only in *one direction*. In order to capture these properties, we provide the adversary in addition to a key generation oracle (as in plain FE) access to additional oracles. Firstly, we allow the adversary to adaptively query corrupted and honest update tokens as well as also provide encryption and honest-ciphertext-update oracles. Furthermore, we want to naturally allow the adversary to see decryption keys for honestly updated challenge ciphertexts.

We show that we can prove our CUFE construction from LWE secure in such a model for the inner-product functionality. Indeed, the tricky part in the proof is to allow the adversary to retrieve functional decryption keys for honestly updated challenge ciphertexts (i.e., it does not see the update token, but has access to an update oracle; see below for detailed discussion). We note that our iO-based construction satisfies the security model for any functionality (see below).

CUFE for any function from (probabilistic) iO. The starting point of our construction is the FE construction due to Garg et al. [GGH+13] which requires iO and uses a public-key encryption (PKE) scheme with the Naor-Yung double encryption paradigm [NY90] using a one-time statistically simulation sound NIZK (1-SSS-NIZK) inside the obfuscated circuit. More precisely, an FE ciphertext is composed of two PKE ciphertexts of a message m under two public keys pk_1 and pk_2, and a NIZK proof attesting that both ciphertexts encrypt the same message. On the other hand, a functional secret key sk_f is the obfuscation of a circuit that has the function f and one of the decryption keys of the PKE hard-coded. Hence, decryption with sk_f amounts to evaluating the obfuscated circuit which in turn verifies the NIZK proof and decrypts the PKE ciphertext to m and then outputs $f(m)$. While proving that the FE adversary cannot distinguish an encryption of m_0 from that of m_1, one could move to an intermediate hybrid experiment where the NIZK proof is simulated, and the two ciphertexts encrypt m_0 and m_1, respectively. However, when simulating a NIZK proof one has to change to a simulated common reference string which implies that valid proofs to false statements exist. Though, iO requires the circuits to be equivalent on all inputs, even on inputs that contain valid proofs for false statements. To overcome this issue, Garg et al. [GGH+13] make use of SSS-NIZK, which requires that except with respect to one particular statement to be simulated, all other valid proofs that exist are only for true statements. We note that the statement must be fixed in advance, thus their construction achieves only selective security.

In order to introduce tags for the ciphertexts, a first step is to replace the 1-SSS-NIZK with one having public labels and to use the tag as a label when computing the proof in the labeled 1-SSS-NIZK. Now the challenging part is to update the ciphertext. In order to restrict that an updated ciphertext cannot be updated anymore, we use two different sets of PKE keys, (sk_o, pk_o) for original ciphertexts and (sk_u, pk_u) for updated ciphertexts. For the update operation, we now need to switch the ciphertext under pk_o (where the NIZK proof carries the original tag as a label) to a new ciphertext under pk_u (where the NIZK proof carries the updated tag as a label).

However, if we want to perform this switching operation in an obfuscated circuit, due to the probabilistic nature of the PKE we need to rely on probabilistic indistinguishability obfuscation (piO) [CLTV15]. The required functionality is then reminiscent of a technique used to obtain universal proxy re-encryption (URE) as proposed by Döttling and Nishimaki [DN21]. Though, in our case we additionally need to compute a fresh NIZK proof with the label representing the new tag inside the obfuscated circuit, which requires a careful design of the class of samplers for piO. We argue that in order to use any IND-CPA secure PKE scheme in our construction, we require stronger security from piO (i.e., we need to assume dynamic-input piO). Alternatively, we can use doubly-probabilistic iO introduced by Agrikola et al. [ACH20], which can be achieved using polynomially secure iO and the exponential DDH assumption.

CUFE for inner-products from standard assumptions. The starting point for the construction from standard assumptions is the identity-based inner-product functional encryption scheme from the LWE assumption by Abdalla et al. [ACGU20]. Their construction essentially combines the LWE-based inner-product FE scheme from Agrawal et al. [ALS16] – we will refer to this scheme as ALS – with a LWE-based IBE scheme, e.g., the IBEs from [GPV08] or [ABB10]. The latter is especially of interest for us: starting from a public key $\mathbf{A}$ it is possible to derive an identity-specific matrix $\mathbf{A}_{id}$ for some identity id. This $\mathbf{A}_{id}$ describes a trapdoor function for which it is hard to compute a short preimage. Yet, given the trapdoor for $\mathbf{A}$, which is stored as part of the main secret key, it is possible to derive sk_{id} as trapdoor for $\mathbf{A}_{id}$. Notably, sk_{id} is a matrix which can be projected to functional decryption keys for inner-products $\langle \cdot, \vec{y} \rangle$, hence giving $sk_{id,\vec{y}}$.

While this idea incidentally gives rise to a tag-based inner-product FE construction, producing update tokens to transform ciphertexts from the source to the target tag is non-obvious. We want to note, however, that this is one of the core challenges that is solved by proxy re-encryption in the public key setting. It is however non-trivial to combine a proxy re-encryption scheme with a functional encryption scheme without running into issues with collusion. Indeed, consider a black-box approach that combines both worlds by encrypting the FE ciphertext with a PRE. Now consider two colluding users t and t' who have functional secret keys for distinct f and f'. Now if a ciphertext is re-encrypted to t, they can use their PRE secret key to remove the PRE layer. Then both t and t' can evaluate their functions by simply sharing the decapsulated FE ciphertext. Therefore, a CUFE scheme requires tighter intertwining of the two concepts to prevent mix-and-match-style and other attacks.

Still, ideas found in lattice-based proxy re-encryption constructions help us to turn ALS combined with tag-based keys into a secure CUFE. We quickly revisit the construction by Fan and Liu [FL19]. Their idea is to set up the user-specific matrices from a global public matrix $\mathbf{A}$. Given such a fixed matrix $\mathbf{A}$, the matrix for a user u is then set to be $\mathbf{A}_u = [\mathbf{A}|\mathbf{A}_{u,1} + \mathbf{H}\mathbf{G}|\mathbf{A}_{u,2} + \mathbf{H}'\mathbf{G}]$ where $\mathbf{A}_{u,i} = -\mathbf{A}\mathbf{R}_{u,i}$ with $\mathbf{R}_{u,i}, i = 1, 2$ contained in the secret key and where $\mathbf{H}'$ is randomly sampled. Encryption follows a dual-Regev approach [GPV08] based on the user dependent matrix $\mathbf{A}_{\mathbf{u}}$. Re-encryption keys for user u to user u' are generated by sampling matrices $\mathbf{X}_{01}, \mathbf{X}_{02}, \mathbf{X}_{11}, \mathbf{X}_{12}$ using $\mathbf{R}_{u,1}, \mathbf{R}_{u,2}$ such that

$$[\mathbf{A}| - \mathbf{A}_{u,1} + h(1)\mathbf{G}| - \mathbf{A}_{u,2} + \mathbf{B}] \begin{bmatrix} \mathbf{I} & \mathbf{X}_{0,1} & \mathbf{X}_{0,2} \\ 0 & \mathbf{X}_{1,1} & \mathbf{X}_{1,2} \\ 0 & 0 & \mathbf{I} \end{bmatrix} = [\mathbf{A}| - \mathbf{A}_{j,1} + h(2)\mathbf{G}| - \mathbf{A}_{j,2} + \mathbf{B}]$$

for any matrix $\mathbf{B}$. In their construction, h is used to describe the "ciphertext level" (either freshly generated, $h(1)$ or updated, $h(2)$) whereas $\mathbf{B}$ stems from a function producing matrices on input of a tag.

The setting of CUFE is however vastly different in nature as ciphertexts are not equipped with levels and there are no per-user public keys. Yet, this method to set up the matrices such that one can update dual-Regev style ciphertexts from one matrix to another is helpful to construct the update tokens. Additionally, with dual-Regev inspired ciphertexts we are also able to set up keys as matrices in such a way that we are able to first sample a tag-specific trapdoor from the main secret key which is then projected to a functional secret key. Consequently, our construction intertwines the functional encryption features from ALS with tag-based ciphertext updates in a non-black-box manner.

As the construction is not black-box, neither is the proof. The main technical challenge in the proof comes from having to produce updates of the challenge ciphertext and function keys for the respective target tags. Embedding an ALS instance (as done for the challenge identity in [ACGU20]) for each of these tags does not work as the different instances should be related in order to simulate the derived matrices of these tags correctly. On the other hand, using a single ALS instance to simulate function keys for multiple tags leads, if done in the trivial way, to producing function keys related to each other, and thus again to a view for the adversary distinguishable from the expected one. However, this drawback can be overcome by "re-randomizing" the function keys in a way that it "hides" the function key provided by the ALS challenger (similarly to Lai et al. [LLW21]). In this way the adversary's view is indistinguishable from that in the real experiment.

1.2 Related Work

While we are not aware of any previous work that tries to achieve the desired goals via ciphertext updatability, a related concept is that of controlled functional encryption (C-FE) [NAP+14]. This approach enhances FE with an authority that needs to be involved in the decryption process and thus allows a fine-grained control over which ciphertexts can be decrypted by a holder of a functional key. Consequently, the access control is enforced by the authority and by dynamically changing which user is allowed to decrypt which ciphertexts one can view this as achieving similar goals as with ciphertext updatability. However, the major difference is that C-FE requires an interactive decryption procedure between the user and authority and thus requires the authority to be online and available all the time. This would potentially hinder scalability in large-scale systems. In contrast, our approach is oblivious to the users. Furthermore, the requirement of an always online authority that needs to be fully trusted might be problematic and undesirable. This trust issue has recently been addressed by distributing the trust in the authority via the concept of Multi-Authority C-FE [AFS21], however, this incurs further communication overhead. Another related (but conceptually different) line of work is updating policies in ABE [Kaw15, FS16]. In general, these works combine ciphertext-policy ABE with PRE in order to update the policy associated with the ciphertext. However, these works neither consider (IP)FE schemes nor are sufficient for our envisioned applications. Our work can be seen as a combination of IBE/ABE with FE augmented by updatability, and, hence, updatability needs to consider and tie both parts together.

2 Preliminaries

Notation. For $n \in \mathbb{N}$, let $[n] := \{1, \ldots, n\}$, and let $\lambda \in \mathbb{N}$ be the security parameter. For a finite set $\mathcal{S}$, we denote by $s \leftarrow \mathcal{S}$ the process of sampling s uniformly from $\mathcal{S}$. Let $y \leftarrow A(\lambda, x)$ be the process of running an algorithm A on input (λ, x) with access to uniformly random coins and assigning the result to y (we may omit to mention the λ-input explicitly and assume that all algorithms take λ as input). To make the random coins r explicit, we write $A(\lambda, x; r)$. We use $\perp$ to indicate that an algorithm terminates with an error and A^B when A has oracle access to B, where B may return $\top$ as a distinguished special symbol. We say an algorithm A is probabilistic polynomial time (PPT) if the running time of A is polynomial in λ. Given $\vec{x} \in \mathbb{Z}^n$, we denote by $\|\vec{x}\|$ its Euclidean norm, i.e., for $\vec{x} = (x_i)_{i\in[n]}$, we have $\|\vec{x}\| := \sqrt{\sum_{i=1}^{n} x_i^2}$. For a matrix $\mathbf{R}$, by $\widetilde{\mathbf{R}}$ we denote the result of applying Gram-Schmidt orthogonalization to the columns of $\mathbf{R}$. By $\|\mathbf{R}\|$, we will denote the Euclidean norm of the longest column of $\mathbf{R}$, and by $s_1(\mathbf{R})$ its spectral norm, i.e., the largest singular value of $\mathbf{R}$. A function f is negligible if its absolute value is smaller than the inverse of any polynomial (i.e., if $\forall c \exists k_0 \forall \lambda \geq k_0 : |f(\lambda)| < 1/\lambda^c$). We may write $q = q(\lambda)$ if we mean that the value q depends polynomially on λ. Given two different distributions X and Y over a countable domain D, we denote their statistical distance as $\mathsf{SD}(X, Y) = \frac{1}{2} \sum_{d \in D} |X(d) - Y(d)|$, and say that X and Y are $\mathsf{SD}(X, Y)$ close.

We recall public-key encryption and non-interactive zero knowledge proof systems in Appendix A.1 and A.2.

2.1 (Probabilistic) Indistinguishability Obfuscation

Definition 1 (Indistinguishability Obfuscator). *A PPT algorithm $i\mathcal{O}$ is an indistinguishability obfuscator (iO) for a circuit class $\{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ if it satisfies the following conditions:*

Experiment $\mathsf{Exp}^{\mathsf{di\text{-}ind}}_{D,A}(\lambda)$
$(C_0, C_1, z) \leftarrow D_\lambda$
$(x, \mathsf{st}) \leftarrow A_1(C_0, C_1, z)$
$y \leftarrow C_b(x)$, for $b \leftarrow_\$ \{0,1\}$
$b' \leftarrow A_2(\mathsf{st}, C_0, C_1, z, x, y)$
if $b = b'$ return 1 else return 0

Fig. 1. Experiment $\mathsf{Exp}^{\mathsf{di\text{-}ind}}$ for the indistinguishability property of dynamic-input samplers.

Functionality. *For any security parameter $\lambda \in \mathbb{N}$, any circuit $C \in \mathcal{C}_\lambda$, and any input x, we have that*

$$\Pr[C'(x) = C(x) \mid C' \leftarrow i\mathcal{O}(C)] = 1.$$

Indistinguishability. *For any PPT distinguisher $\mathcal{D}$ and for any pair of circuits $C_0, C_1 \in \mathcal{C}_\lambda$, such that for any input x, $C_0(x) = C_1(x)$ and $|C_0| = |C_1|$, it holds that*

$$|\Pr[\mathcal{D}(i\mathcal{O}(C_0)) = 1] - \Pr[\mathcal{D}(i\mathcal{O}(C_1)) = 1]| \leq \mathsf{negl}(\lambda).$$

We further say that $i\mathcal{O}$ is subexponentially secure if for any PPT $\mathcal{D}$ the above advantage is smaller than $2^{-\lambda^\varepsilon}$ for some $0 < \varepsilon < 1$.

Next, we consider a family of sets of randomized polynomial-size circuits $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$. We define a circuit sampler for $\mathcal{C}$ as a distribution ensemble $D = \{D_\lambda\}_{\lambda\in\mathbb{N}}$, where the distribution of D is (C_0, C_1, z) with $C_0, C_1 \in \mathcal{C}_\lambda$ such that C_0 and C_1 take inputs of the same length, and $z \in \{0,1\}^{\mathsf{poly}(\lambda)}$ is the auxiliary information. A class $\mathcal{S}$ of samplers for the circuit family $\mathcal{C}$ is a set of circuit samplers for $\mathcal{C}$.

Definition 2 (piO for a Class of Samplers [CLTV15, DHRW16]). *A PPT algorithm $pi\mathcal{O}$ is a probabilistic indistinguishability obfuscator (piO) for a class of samplers $\mathcal{S}$ over the randomized circuit family $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ if it satisfies the following conditions:*

Correctness. *For any security parameter $\lambda \in \mathbb{N}$, any $C \in \mathcal{C}_\lambda$, and any input x, the distributions of $C(x)$ over the random coins of C and of $C' \leftarrow pi\mathcal{O}(1^\lambda, C)$ over the random coins of the obfuscator are identical.*

Security with respect to $\mathcal{S}$. *For every sampler $D = \{D_\lambda\}_{\lambda\in\mathbb{N}} \in \mathcal{S}$, and for every non-uniform PPT machine A, there exists a negligible function* negl*, such that*

$$\Big| \Pr\left[(C_0, C_1, z) \leftarrow D_\lambda \colon A(C_0, C_1, pi\mathcal{O}(1^\lambda, C_0), z) = 1\right] \\ - \Pr\left[(C_0, C_1, z) \leftarrow D_\lambda \colon A(C_0, C_1, pi\mathcal{O}(1^\lambda, C_1), z) = 1\right] \Big| \leq \mathsf{negl}(\lambda).$$

Canetti et al. [CLTV15] defined four types of samplers, from which we only review the dynamic-input indistinguishable sampler here. Roughly, a dynamic-input indistinguishability sampler is required to output circuits $C_0, C_1 \in \mathcal{C}_\lambda$, such that the output of these circuits on a dynamically chosen input is computationally indistinguishable.

Definition 3 (Dynamic-input Indistinguishability Sampler [CLTV15]). *The class $\mathcal{S}^{\mathsf{di\text{-}ind}}$ of dynamic-input samplers for a circuit family $\mathcal{C}$ contains all circuit samplers $D = \{D_\lambda\}_{\lambda\in\mathbb{N}}$ for $\mathcal{C}$ satisfying the following: for every non-uniform PPT $A = (A_1, A_2)$, it holds that*

$$\mathsf{Adv}^{\mathsf{di\text{-}ind}}_{D,A}(\lambda) := \left|\Pr\left[\mathsf{Exp}^{\mathsf{di\text{-}ind}}_{D,A}(\lambda) = 1\right] - 1/2\right|,$$

in the experiment $\mathsf{Exp}^{\mathsf{di\text{-}ind}}_{D,A}$ represented in Fig. 1 is negligible.

We note that the dynamic-input piO is the strongest notion defined in [CLTV15] and corresponds to a randomized variant of differing-input obfuscation [BGI+12]. Hence, it inherits the implausibility results of differing-input obfuscation for general circuits [GGHW14, BSW16]. However, Canetti et al. [CLTV15] argued that a construction of dynamic-input piO for specific classes of samplers is possible, analogous to the case of differing-input obfuscation for specific circuits.

3 Ciphertext-Updatable Functional Encryption

We present our definitional framework of ciphertext-updatable functional encryption (CUFE). CUFE is a tag-based functional-encryption (FE) scheme defined on functionality $\mathcal{F}\colon \mathcal{X} \to \mathcal{Y}$ and tag space $\mathcal{T}$. Key generation outputs a main public-secret key pair (mpk, msk), where from msk, the function keys $sk_{f,t}$ for some function $f \in \mathcal{F}$ and tag $t \in \mathcal{T}$ can be derived. Encryption is done according to some tag $t \in \mathcal{T}$ and message $x \in \mathcal{X}$. Now, if the tag of the function key and the ciphertext match, then decryption succeeds and outputs $f(x)$. Furthermore, we want to allow switching of tags, i.e., from t to t', in a ciphertext once, which is carried out via tokens $\Delta_{t \to t'}$. Such a token can be used to update a ciphertext C_t to a ciphertext $C_{t'}$ under the tag t' specified in the token but not vice versa, i.e., from t' to t.

Definition 4. *A CUFE scheme* CUFE *for functionality* $\mathcal{F}\colon \mathcal{X} \to \mathcal{Y}$ *with message space* $\mathcal{X}$ *and tag space* $\mathcal{T}$ *is a tuple of the PPT algorithms:*

$\mathsf{Setup}(\lambda, \mathcal{F})$: *on input security parameter* $\lambda \in \mathbb{N}$ *and a class of functions* $\mathcal{F}$*, the setup algorithm outputs a main public-secret key pair* (mpk, msk)*.*

$\mathsf{KeyGen}(msk, f, t)$: *on input* msk*, function* $f \in \mathcal{F}$*, and tag* $t \in \mathcal{T}$*, the key-generation algorithm outputs a function key* $sk_{f,t}$*.*

$\mathsf{TokGen}(msk, t, t')$: *on input* msk *and tags* $t, t' \in \mathcal{T}$*, the token-generation algorithm outputs an update token* $\Delta_{t \to t'}$*.*

$\mathsf{Enc}(mpk, x, t)$: *on input* mpk*, message* $x \in \mathcal{X}$*, and tag* $t \in \mathcal{T}$*, the encryption algorithm outputs a ciphertext* C_t *for* x*.*

$\mathsf{Update}(\Delta_{t \to t'}, C_t)$: *on input an update token* $\Delta_{t \to t'}$ *and ciphertext* C_t*, the update algorithm outputs an updated ciphertext* $UC_{t'}$ *or* $\perp$*.*

$\mathsf{Dec}(sk_{f,t'}, C_t/UC_t)$[8]: *on input function key* $sk_{f,t'}$ *and a ciphertext (either a non-updated one* C_t *or an updated one* UC_t*), the decryption algorithm outputs* $f(x) \in \mathcal{Y}$ *if* $t' = t$*, else outputs* $\perp$*.*

Correctness for CUFE. *Correctness essentially guarantees that if the tag in a function key and in an (updated) ciphertext match, then decryption succeeds.*

More concretely, a CUFE scheme CUFE *is correct if for all* $\lambda \in \mathbb{N}$*, for any* $\mathcal{F}\colon \mathcal{X} \to \mathcal{Y}$*, for any* $(mpk, msk) \leftarrow \mathsf{Setup}(\lambda, \mathcal{F})$*, for any* $f \in \mathcal{F}$*, for any* $t \in \mathcal{T}$*, for any* $sk_{f,t} \leftarrow \mathsf{KeyGen}(msk, f, t)$*, for any* $x \in \mathcal{X}$*, for any* $C_t \leftarrow \mathsf{Enc}(mpk, x, t)$*, we have that* $\mathsf{Dec}(sk_{f,t}, C_t) = f(x)$ *holds, and for any* $t' \in \mathcal{T} \setminus \{t\}$*, for any* $\Delta_{t \to t'} \leftarrow \mathsf{TokGen}(msk, t, t')$*, for any* $UC_{t'} \leftarrow \mathsf{Update}(\Delta_{t \to t'}, C_t)$*, we have that* $\mathsf{Dec}(sk_{f,t'}, UC_{t'}) = f(x)$ *holds.*

Remark 1. Notice that the correctness of the CUFE scheme only guarantees that non-updated ciphertexts for tag t can be updated to tag t' using the update token $\Delta_{t \to t'}$ and still be decrypted correctly. Looking ahead to the CPA security notion, this will be the only possible use of the update token. Any other successful use (e.g., updating ciphertexts in the reverse direction or updating already updated ciphertexts) will allow the adversary to win the security experiment (see below). Hence, a secure CUFE construction implies that the update token can *only* be used to update a non-updated ciphertext to an updated one (assuming the tags match), but not vice versa and not multiple times (i.e., to "update" an already updated ciphertext is not possible as this would penalize CUFE security).

Intuition of our CPA security notions for CUFE. Updating ciphertexts via tokens is closely related to the realm of proxy re-encryption (PRE) [BBS98, AFGH05] and, indeed, we start from the recent PRE state-of-the-art security model by Cohen [Coh19] and carefully adapt such a model to our needs in the chosen-plaintext-attack indistinguishability setting. Moreover, due to the updatability of ciphertexts and thus the concept of update tokens not being present in plain FE, we need to require additional aspects for our security guarantees. Such tokens could potentially be used to also switch function keys or even invert updated ciphertexts. In that vein, we define an indistinguishability-based notion, we dub $\mathsf{IND\text{-}CUFE\text{-}CPA}$, which guarantees that an adversary cannot distinguish ciphertexts for a certain target tag t^* and adversarially chosen messages (x_0^*, x_1^*).

We only want to allow updating the tags of ciphertexts via the token, only in one direction, and only from non-updated to updated ciphertexts. In order to capture these properties, we provide

[8] The decryption algorithms takes either a non-updated ciphertext or an updated one but not both. We assume that one can retrieve the information on the update status from the ciphertexts efficiently.

Experiment $\mathsf{Exp}^{\text{ind-cufe-cpa}}_{\mathsf{CUFE},A}(\lambda, \mathcal{F})$
$(mpk, msk) \leftarrow \mathsf{Setup}(\lambda, \mathcal{F})$
$\mathcal{K} := \emptyset, \mathcal{C} := \mathcal{UC} := \emptyset, \mathcal{HT} := \mathcal{CT} := \emptyset, \mathsf{c} := \mathsf{uc} := 1, \mathsf{ht} := \mathsf{ct} := 1$
$(t^*, x_0^*, x_1^*, \mathsf{st}) \leftarrow A^{\mathcal{O}_1}(mpk)$
$b \leftarrow \{0,1\}$
$C^* \leftarrow \mathsf{Enc}(mpk, x_b, t^*)$
$\mathcal{C} := \mathcal{C} \cup \{(0, C^*, t^*)\}$
$b' \leftarrow A^{\mathcal{O}}(C^*, \mathsf{st})$
if $b' = b$ and A is *valid* then return 1 else return 0

Oracles $\mathcal{O}$
$\mathsf{KeyGen}'(f, t)$: If $f \notin \mathcal{F}$ or $t \notin \mathcal{T}$, then return $\perp$. Compute $sk_{f,t} \leftarrow \mathsf{KeyGen}(msk, f, t)$, set $\mathcal{K} := \mathcal{K} \cup \{(f, t)\}$ and return $sk_{f,t}$.
$\mathsf{HonTokGen}(t, t')$: If $t' \notin \mathcal{T}$, $t \notin \mathcal{T}$, or $(\cdot, t, t') \in \mathcal{CT}$, then return $\perp$. Compute $\Delta_{t \to t'} \leftarrow \mathsf{TokGen}(msk, t, t')$ and set $\mathcal{HT} := \mathcal{HT} \cup \{(\mathsf{ht}, t, t', \Delta_{t \to t'})\}$, $\mathsf{ht} := \mathsf{ht} + 1$.
$\mathsf{CorTokGen}(t, t')$: If $t' \notin \mathcal{T}$, $t \notin \mathcal{T}$, or $(\cdot, t, t', \cdot) \in \mathcal{HT}$, then return $\perp$. Compute $\Delta_{t \to t'} \leftarrow \mathsf{TokGen}(msk, t, t')$, set $\mathcal{CT} := \mathcal{CT} \cup \{(\mathsf{ct}, t, t')\}$, $\mathsf{ct} := \mathsf{ct} + 1$, and return $\Delta_{t \to t'}$.
$\mathsf{Enc}'(x, t)$: Compute $C_t \leftarrow \mathsf{Enc}(mpk, x, t)$, set $\mathcal{C} := \mathcal{C} \cup \{(\mathsf{c}, C_t, t)\}$ and $\mathsf{c} := \mathsf{c} + 1$, and return C_t.
$\mathsf{HonUpdate}(t, t', i, j)$: If $(i, \cdot, t) \notin \mathcal{C}$ or $(\cdot, t, t') \in \mathcal{CT}$, then return $\perp$. If $(j, t, t', \cdot) \notin \mathcal{HT}$, compute $\Delta_{t \to t'} \leftarrow \mathsf{TokGen}(msk, t, t')$ and set $\mathcal{HT} := \mathcal{HT} \cup \{\mathsf{ht}, t, t', \Delta_{t \to t'}\}$, $\mathsf{ht} := \mathsf{ht} + 1$; otherwise, retrieve $(j, t, t', \Delta_{t \to t'})$ from $\mathcal{HT}$. Retrieve (i, C_t, t) from $\mathcal{C}$ and compute $UC_{t'} \leftarrow \mathsf{Update}(\Delta_{t \to t'}, C_t)$. Set $\mathcal{UC} := \mathcal{UC} \cup \{(\mathsf{uc}, i, t')\}$, $\mathsf{uc} := \mathsf{uc} + 1$, and return $UC_{t'}$.

Validity of A
An adversary A is *valid* if and only if:
a) there is no $(f, t^*) \in \mathcal{K}$ with $f(x_0^*) \neq f(x_1^*)$ (i.e., the adversary cannot trivially distinguish the challenge ciphertext),
b) there is no $(f, t) \in \mathcal{K}$ with $f(x_0^*) \neq f(x_1^*)$ and $(\cdot, t^*, t) \in \mathcal{CT}$ (i.e., the adversary has not received update tokens towards t for the challenge ciphertexts where it has queried function keys under t with $f(x_0^*) \neq f(x_1^*)$),
c) there is no $(\cdot, 0, t') \in \mathcal{UC}$ for which $(f, t') \in \mathcal{K}$ exists with $f(x_0^*) \neq f(x_1^*)$ (i.e., the adversary has only queried updated challenge ciphertexts for which it has function keys that satisfy $f(x_0^*) = f(x_1^*)$).

Fig. 2. The $\mathsf{IND\text{-}CUFE\text{-}CPA}$ security notion for CUFE. If $\mathcal{O}_1 = \perp$ we call the security game selective and if $\mathcal{O}_1 = \mathcal{O}$ we call it adaptive.

the adversary in addition to KeyGen (as in plain FE) access to four more oracles. Two of those additional oracles are related to the generation of tokens and the other two are needed to ensure security related to updatability of honestly generated ciphertexts.

Concerning the oracles for the token generation, we allow the adversary to adaptively query corrupted tokens via $\mathsf{CorTokGen}$ and honest tokens via $\mathsf{HonTokGen}$. The former mirrors attacks where the adversary gets complete control over tokens while the latter allows the adversary to query the generation of an honest token without access to the token itself.

Moreover, we also provide Enc' and $\mathsf{HonUpdate}$ oracles. Thereby, Enc' allows generating honest ciphertexts (under mpk) and $\mathsf{HonUpdate}$ allows updating ciphertexts which have been honestly generated via Enc' without revealing the update token to the adversary. See that via $\mathsf{HonTokGen}$, the adversary can query an honest token generation and the experiment can use such a token for the honest update.

The validity of the adversary is checked in the end of the security game. Essentially, the adversary is valid if and only if:

a) the adversary cannot trivially distinguish the challenge ciphertext,
b) the adversary has not received update tokens towards t for the challenge ciphertexts where it has queried function keys under t with $f(x_0^*) \neq f(x_1^*)$,
c) the adversary has only queried updated challenge ciphertexts for which it has function keys that satisfy $f(x_0^*) = f(x_1^*)$.

If the adversary is valid and it has correctly guessed which message was encrypted in the challenge ciphertext, the adversary wins the game.

IND-CUFE-CPA security. We say that a CUFE scheme is IND-CUFE-CPA-secure if any PPT adversary succeeds in the following experiment only with probability negligibly larger than 1/2. The experiment starts by computing the initial main public and secret key pair $(mpk, msk) \leftarrow \mathsf{Setup}(\lambda, \mathcal{F})$, initializes empty sets $\mathcal{K}$, $\mathcal{C}$, $\mathcal{UC}$, $\mathcal{HT}$, $\mathcal{CT}$ to track keys, ciphertexts, updated ciphertexts, honest and corrupted tokens, respectively, as well as initializes the counters c, uc, ht, ct for ciphertexts, updated ciphertexts, honest tokens and corrupted tokens, respectively.

At some point, the adversary outputs target tag and messages (t^*, x_0^*, x_1^*). Next, the experiment tosses a coin b, computes $C^* \leftarrow \mathsf{Enc}(mpk, x_b^*, t^*)$, adds $(0, C^*, t^*)$ to $\mathcal{C}$, and gives C^* to the adversary. The adversary eventually outputs a guess b', where the experiment returns 1 if $b' = b$ and the adversary is valid. In the adaptive security game the adversary has full access to all oracles from the beginning, whereas in the selective security game the adversary only gets access to the oracles after committing to the target tag t^* and challenge messages (x_0^*, x_1^*). Figure 2 depicts the experiment.

Definition 5 (IND-CUFE-CPA security). *A CUFE scheme* CUFE *is* IND-CUFE-CPA*-secure iff for any valid PPT adversary A the advantage function*

$$\mathsf{Adv}^{\mathsf{ind\text{-}cufe\text{-}cpa}}_{\mathsf{CUFE},A}(\lambda, \mathcal{F}) := \left| \Pr\left[\mathsf{Exp}^{\mathsf{ind\text{-}cufe\text{-}cpa}}_{\mathsf{CUFE},A}(\lambda, \mathcal{F}) = 1 \right] - 1/2 \right|,$$

is negligible in λ, where $\mathsf{Exp}^{\mathsf{ind\text{-}cufe\text{-}cpa}}_{\mathsf{CUFE},A}$ is defined in Figure 2.

Remark 2. We model selective (i.e., the target tag and messages are chosen by the adversary before it has access to oracles) as well as adaptive (i.e., the adversary has access to the oracles before specifying target tag and messages) security. We note that it would also be possible to define either only the tag or the messages in a selective sense. This is straightforward to model and we omit it for the sake of simplicity. Also, we note that moving a selective setting to an adaptive one can be done by the standard technique of complexity leveraging if one is willing to accept that message and/or tag spaces are polynomially bounded in the security parameter.

4 Generic Construction of CUFE and Extensions

In this section, we present a generic construction of CUFE for any function from indistinguishability obfuscation that provides full IND-CUFE-CPA security. For the sake of consistency, we opt to present it for the equality predicate on tags and then extend the expressiveness of predicates beyond the equality testing on tags. We show that due to the way our construction is built, it easily allows to support any predicate that can be represented as a circuit of arbitrary polynomial size. Moreover, we remark that one can obtain adaptive FE security using the black-box transformation of Ananth et al. [ABSV15] along with applying complexity leveraging over the tag space.

4.1 Generic CUFE from iO for any Function

The generic construction is inspired by the approach to construct functional encryption from indistinguishability obfuscation by Garg et al. [GGH+13]. Our construction uses indistinguishability obfuscator $i\mathcal{O}$, probabilistic indistinguishability obfuscator $pi\mathcal{O}$, public-key encryption scheme Σ, and labeled statistically simulation sound non-interactive zero-knowledge (ℓ-SSS-NIZK) proof system Π. The construction is described below (where the parts in blue in programs PKey:2 and PUpdate:2 highlight the changes with respect to programs PKey:1 and PUpdate:1):

– $\mathsf{Setup}(1^\lambda, \mathcal{F})$: Compute the following:
  1. Generate $(sk_{o,1}, pk_{o,1}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\lambda)$ and $(sk_{o,2}, pk_{o,2}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\lambda)$.
  2. Generate $(sk_{u,1}, pk_{u,1}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\lambda)$ and $(sk_{u,2}, pk_{u,2}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\lambda)$.
  3. Set $\mathsf{crs} \leftarrow \Pi.\mathsf{Setup}(1^\lambda)$.

  Output the main public/secret key pair $(mpk := (pk_{o,1}, pk_{o,2}, pk_{u,1}, pk_{u,2}, \mathsf{crs}), msk := (sk_{o,1}, sk_{u,1}))$.
– $\mathsf{KeyGen}(msk, f, t)$: Compute an obfuscation $P_{f,t} \leftarrow i\mathcal{O}(\text{PKey:1}[f, t, sk_{o,1}, sk_{u,1}, \mathsf{crs}])$ for the program PKey:1$[f, t, sk_{o,1}, sk_{u,1}, \mathsf{crs}]$ with the circuit size equal to $\max\{|\text{PKey:1}[f, tsk_{o,1}, sk_{u,1}, \mathsf{crs}]|, |\text{PKey:2}[f, t, sk_{o,2}, sk_{u,2}, \mathsf{crs}]|\}$. Output the secret key $sk_{f,t} := P_{f,t}$.

PKey:1

Constants: $f, t, sk_{o,1}, sk_{u,1}, \mathsf{crs}$
Inputs: $C_t := (e_1, e_2, \pi_t)$

1. If C_t is updated ciphertext, set $(pk_1, pk_2, sk_2) := (pk_{u,1}, pk_{u,2}, sk_{u,1})$, else set $(pk_1, pk_2, sk_2) := (pk_{o,1}, pk_{o,2}, sk_{o,1})$.
2. If $\Pi.\mathsf{Verify}(\mathsf{crs}, t, x, \pi_t) \neq 1$, for $x := \{\exists m, r_1, r_2 \mid e_1 = \Sigma.\mathsf{Enc}(pk_1, m; r_1) \wedge e_2 = \Sigma.\mathsf{Enc}(pk_2, m; r_2)\}$, output $\perp$.
3. Else output $f(\Sigma.\mathsf{Dec}(sk_1, e_1))$.

PKey:2

Constants: $f, t, sk_{o,2}, sk_{u,2}, \mathsf{crs}$
Inputs: $C_t := (e_1, e_2, \pi_t)$

1. If C_t is updated ciphertext, set $(pk_1, pk_2, sk_2) := (pk_{u,1}, pk_{u,2}, sk_{u,2})$, else set $(pk_1, pk_2, sk_2) := (pk_{o,1}, pk_{o,2}, sk_{o,2})$.
2. If $\Pi.\mathsf{Verify}(\mathsf{crs}, t, x, \pi_t) \neq 1$, for $x := \{\exists m, r_1, r_2 \mid e_1 = \Sigma.\mathsf{Enc}(pk_1, m; r_1) \wedge e_2 = \Sigma.\mathsf{Enc}(pk_2, m; r_2)\}$, output $\perp$.
3. Else output $f(\Sigma.\mathsf{Dec}(sk_2, e_2))$.

PUpdate:1

Constants: $t, t', sk_{o,1}, mpk := (pk_{o,1}, pk_{o,2}, pk_{u,1}, pk_{u,2}, \mathsf{crs})$
Inputs: $C_t := (e_1, e_2, \pi_t)$

1. If $\Pi.\mathsf{Verify}(\mathsf{crs}, t, x, \pi_t) \neq 1$, for $x := \{\exists m, r_1, r_2 \mid e_1 = \Sigma.\mathsf{Enc}(pk_{o,1}, m; r_1) \wedge e_2 = \Sigma.\mathsf{Enc}(pk_{o,2}, m; r_2)\}$, output $\perp$.
2. Compute $m \leftarrow \Sigma.\mathsf{Dec}(sk_{o,1}, e_1)$, and if $m = \perp$ output $\perp$.
3. Compute $e'_1 \leftarrow \Sigma.\mathsf{Enc}(pk_{u,1}, m; r'_1)$ and $e'_2 \leftarrow \Sigma.\mathsf{Enc}(pk_{u,2}, m; r'_2)$.
4. Compute $\pi_{t'} \leftarrow \Pi.\mathsf{Prove}(\mathsf{crs}, t', x, w)$, where t' is the label, $w := (m, r'_1, r'_2)$ and $x := \{\exists m, r'_1, r'_2 \mid e'_1 = \Sigma.\mathsf{Enc}(pk_{u,1}, m; r'_1) \wedge e'_2 = \Sigma.\mathsf{Enc}(pk_{u,2}, m; r'_2)\}$.
5. Output $C_{t'} := (e'_1, e'_2, \pi_{t'})$.

- $\mathsf{TokGen}(msk, t, t')$: Compute an obfuscation $P_{t \to t'} \leftarrow pi\mathcal{O}(\text{PUpdate:1}[t, t', sk_{o,1}, mpk])$ for the program $\text{PUpdate:1}[t, t', sk_{o,1}, mpk]$ with the circuit size equal to $\max\{|\text{PUpdate:1}[t, t', sk_{o,1}, mpk]|, |\text{PUpdate:2}[t, t', sk_{o,2}, mpk]|\}$. Output the update token $\Delta_{t \to t'} := P_{t \to t'}$.
- $\mathsf{Enc}(mpk, m, t)$: Compute the following:
  1. $e_1 \leftarrow \Sigma.\mathsf{Enc}(pk_{o,1}, m; r_1)$ and $e_2 \leftarrow \Sigma.\mathsf{Enc}(pk_{o,2}, m; r_2)$.
  2. $\pi_t \leftarrow \Pi.\mathsf{Prove}(\mathsf{crs}, t, x, w)$, where t is the label, $w := (m, r_1, r_2)$ is a witness for the NP statement

$$x := \{\exists m, r_1, r_2 \mid e_1 = \Sigma.\mathsf{Enc}(pk_{o,1}, m; r_1) \wedge e_2 = \Sigma.\mathsf{Enc}(pk_{o,2}, m; r_2)\}.$$

  Output the ciphertext $C_t := (e_1, e_2, \pi_t)$.
- $\mathsf{Update}(\Delta_{t \to t'} := P_{t \to t'}, C_t)$: Run the obfuscated program $C_{t'} \leftarrow P_{t \to t'}(C_t)$ and output $C_{t'}$.
- $\mathsf{Dec}(sk_{f,t} := P_{f,t}, C_t)$: Run the obfuscated program $f(m) \leftarrow P_{f,t}(C_t)$ and output $f(m)$.

Correctness. The correctness of our construction follows straightforwardly from the correctness of the obfuscators $i\mathcal{O}$ and $pi\mathcal{O}$, public-key encryption scheme Σ, ℓ-SSS-NIZK proof system Π, and the description of the programs PKey:1 and PUpdate:1.

Security Proof. Towards establishing the security of our generic construction, we rely on the security properties of the obfuscators $i\mathcal{O}$ and $pi\mathcal{O}$. We assume that $i\mathcal{O}$ is an indistinguishability obfuscator for the circuit class $\mathcal{C}_\lambda$ and that PKey:1, PKey:2 $\in \mathcal{C}_\lambda$. Furthermore, we assume that $pi\mathcal{O}$ is a probabilistic indistinguishability obfuscator for the class of samplers $\mathcal{S}^{\Sigma,\Pi}$, which are defined by the public-key encryption scheme Σ and ℓ-SSS-NIZK proof system Π. The samplers in $\mathcal{S}^{\Sigma,\Pi}$ sample pair of circuits that correspond to the circuits used in our generic construction. We describe the class of samplers in Fig. 3.

PUpdate:2

Constants: $t, t', sk_{o,2}, mpk := (pk_{o,1}, pk_{o,2}, pk_{u,1}, pk_{u,2}, \mathsf{crs})$
Inputs: $C_t := (e_1, e_2, \pi_t)$

1. If $\Pi.\mathsf{Verify}(\mathsf{crs}, t, x, \pi_t) \neq 1$, for $x := \{\exists m, r_1, r_2 \mid e_1 = \Sigma.\mathsf{Enc}(pk_{o,1}, m; r_1) \wedge e_2 = \Sigma.\mathsf{Enc}(pk_{o,2}, m; r_2)\}$, output $\perp$.
2. Compute $m \leftarrow \Sigma.\mathsf{Dec}(sk_{o,2}, e_2)$, and if $m = \perp$ output $\perp$.
3. Compute $e'_1 \leftarrow \Sigma.\mathsf{Enc}(pk_{u,1}, m; r'_1)$ and $e'_2 \leftarrow \Sigma_u.\mathsf{Enc}(pk_{u,2}, m; r'_2)$.
4. Compute $\pi_{t'} \leftarrow \Pi.\mathsf{Prove}(\mathsf{crs}, t', x, w)$, where t' is the label, $w := (m, r'_1, r'_2)$ and $x := \{\exists m, r'_1, r'_2 \mid e'_1 = \Sigma.\mathsf{Enc}(pk_{u,1}, m; r'_1) \wedge e'_2 = \Sigma.\mathsf{Enc}(pk_{u,2}, m; r'_2)\}$.
5. Output $C_{t'} := (e'_1, e'_2, \pi_{t'})$.

$\Sigma = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is a public-key encryption scheme, $\Pi = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ is an ℓ-SSS-NIZK proof system and $\mathcal{T} = \{(t, t')\}$ and $\mathcal{K} = \{(sk, pk)\}$ are sequences of pairs of strings of length $t(\lambda)$ and $k(\lambda)$, respectively.

Sampler $D^{\mathcal{T},\mathcal{K}}$**:** The distribution $D^{\mathcal{T},\mathcal{K}}$ samples $\mathsf{crs} \leftarrow \Pi.\mathsf{Setup}(1^\lambda)$, and outputs $C_0 =$ PUpdate:1$[t, t', sk_{o,1}, mpk]$, $C_1 =$ PUpdate:2$[t, t', sk_{o,2}, mpk]$ and $z = (t, t', mpk)$, where $mpk = (pk_{o,1}, pk_{o,2}, pk_{u,1}, pk_{u,2}, \mathsf{crs})$, and $(t, t') \in \mathcal{T}$ and $(sk_{o,1}, pk_{o,1}), (sk_{o,2}, pk_{o,2}), (sk_{u,1}, pk_{u,1}), (sk_{u,2}, pk_{u,2}) \in \mathcal{K}$.

Class $\mathcal{S}^{\Sigma,\Pi}$**:** Let $\mathcal{S}^{\Sigma,\Pi}$ be the class of samplers with distribution $D^{\mathcal{T},\mathcal{K}}$ for all sequences of strings $\mathcal{T}$ and $\mathcal{K}$.

Fig. 3. The class of samplers for proving the security of our generic construction.

Next, we present the proof of IND-CUFE-CPA security of our generic construction.

Theorem 1. *Let Σ be an* IND-CPA *secure public-key encryption scheme, Π be an (one-time) ℓ-SSS-NIZK proof system, $i\mathcal{O}$ be an indistinguishability obfuscator for the circuit class $\mathcal{C}_\lambda$ and $pi\mathcal{O}$ be a probabilistic indistinguishability obfuscator for the class of samplers $\mathcal{S}^{\Sigma,\Pi}$. Then, our generic construction is a selectively* IND-CUFE-CPA *secure CUFE scheme.*

Proof. For simplicity, we assume that a poly-time adversary A makes exactly Q_k function secret key queries and $Q_t = Q_{ht} + Q_{ct}$ token generation queries (where Q_{ht} and Q_{ct} denote the number of honest and corrupted token generation queries, respectively). We denote by f_i, for $i \in [Q_k]$, the i-th function queried to the secret key generation oracle. We use (t_i, t'_i), for $i \in [Q_t]$, to denote the i-th tag pair queried to the token generation oracles. The proof is organized in a sequence of hybrid experiments, where initially the challenger encrypts m_0 and we gradually (in multiple hybrid steps) change the encryption into an encryption of m_1. Below we formally describe all the hybrids, and hereafter, let $\mathsf{Hybrid}_i \approx \mathsf{Hybrid}_{i+1}$ denote $\left|\Pr[\mathsf{Hybrid}_i = 1] - \Pr[\mathsf{Hybrid}_{i+1} = 1]\right| \leq \mathsf{negl}(\lambda)$.

– Hybrid_0: This hybrid corresponds to the honest execution of the selective variant of indistinguishability game given in Section 3, such that the adversary selects a challenge tag t^* and the challenger encrypts m_0 in the challenge ciphertext.
– Hybrid_1: This hybrid is identical to Hybrid_0 with the exception that $(\mathsf{crs}, \pi_{t^*})$ is simulated as

$$(\mathsf{crs}, \pi_{t^*}) \leftarrow \Pi.\mathsf{Sim}(1^\lambda, t^*, \{\exists m, r_1, r_2 \mid e^*_1 \leftarrow \Sigma.\mathsf{Enc}(pk_{o,1}, m; r_1) \\ \wedge e^*_2 \leftarrow \Sigma.\mathsf{Enc}(pk_{o,2}, m; r_2)\}),$$

where e^*_1 and e^*_2 are part of the challenge ciphertext C_{t^*}.
– Hybrid_2: This hybrid is identical to Hybrid_1 with the exception that the challenge ciphertext is generated as $C_{t^*} = (e^*_1 := \Sigma.\mathsf{Enc}(pk_{o,1}, m_0; r_1), e^*_2 := \Sigma.\mathsf{Enc}(pk_{o,2}, m_1; r_2), \pi_{t^*})$, where π_{t^*} is still simulated.
– $\mathsf{Hybrid}_{3,i}$ for $i \in [0, Q_k]$: In $\mathsf{Hybrid}_{3,i}$ the first i function secret keys queries are answered with the obfuscation of the program PKey:2$[f_i, t, sk_{o,2}, sk_{u,2}, \mathsf{crs}]$, and the remaining $(i + 1)$ to Q_k queries are answered using the program PKey:1$[f_i, t, sk_{o,1}, sk_{u,1}, \mathsf{crs}]$. We note that $\mathsf{Hybrid}_{3,0}$ is equivalent to Hybrid_2.

– $\mathsf{Hybrid}_{4,i}$ for $i \in [0, Q_t]$: In $\mathsf{Hybrid}_{4,i}$ the first i token queries are answered with the obfuscation of the program PUpdate:2$[t_i, t'_i, sk_{o,2}, mpk]$, and the remaining $(i+1)$ to Q_t queries are answered using the program PUpdate:1$[t_i, t'_i, sk_{o,1}, mpk]$. We note that $\mathsf{Hybrid}_{4,0}$ is equivalent to Hybrid_{3,Q_k}.
– Hybrid_5: This hybrid is identical to Hybrid_{4,Q_t} with the exception that the challenge ciphertext is generated as $C_{t^*} = (e_1^* := \Sigma.\mathsf{Enc}(pk_{o,1}, m_1; r_1), e_2^* := \Sigma.\mathsf{Enc}(pk_{o,2}, m_1; r_2), \pi_{t^*})$, where π_{t^*} is still simulated.
– $\mathsf{Hybrid}_{6,i}$ for $i \in [0, Q_t]$: The challenge ciphertext and CRS remains as in Hybrid_5. Otherwise, in $\mathsf{Hybrid}_{6,i}$ the first i token queries are answered with the obfuscation of the program PUpdate:1$[t_i, t'_i, sk_{o,1}, mpk]$, and the remaining $(i+1)$ to Q_t queries are answered using the program PUpdate:2$[t_i, t'_i, sk_{o,2}, mpk]$ as in Hybrid_5. We note that $\mathsf{Hybrid}_{6,0}$ is equivalent to Hybrid_5.
– $\mathsf{Hybrid}_{7,i}$ for $i \in [0, Q_k]$: In $\mathsf{Hybrid}_{7,i}$ the first i function secret keys queries are answered with the obfuscation of the program PKey:1$[f_i, t, sk_{o,1}, sk_{u,1}, \mathsf{crs}]$, and the remaining $(i+1)$ to Q_k queries are answered using the program PKey:2$[f_i, t, sk_{o,2}, sk_{u,2}, \mathsf{crs}]$. We note that $\mathsf{Hybrid}_{7,0}$ is equivalent to Hybrid_{6,Q_t}.
– Hybrid_8: This hybrid is identical to Hybrid_{7,Q_k} with the exception that the CRS and the proof π_{t^*} are generated honestly (using $\Pi.\mathsf{Setup}$ and the witness (r_1, r_2), respectively). This corresponds to the security game where the message m_1 is encrypted for the challenge ciphertext.

Lemma 1. *If ℓ-SSS-NIZK proof system Π is computationally zero-knowledge, then it holds that* $\mathsf{Hybrid}_0 \approx \mathsf{Hybrid}_1$.

Proof. We construct an adversary B for the zero-knowledge property of Π. First, B is given a target tag t^* and a pair of messages m_0, m_1 by A. Next, B generates all public/secret keys that form mpk and msk, and computes the encryptions $e_1^* \leftarrow \Sigma.\mathsf{Enc}(pk_{o,1}, m_0; r_1)$ and $e_2^* \leftarrow \Sigma.\mathsf{Enc}(pk_{o,2}, m_0; r_2)$. Then, B submits to the zero-knowledge challenger of Π the statement

$$x' := \{\exists m, r_1, r_2 \mid e_1^* \leftarrow \Sigma.\mathsf{Enc}(pk_{o,1}, m; r_1) \wedge e_2^* \leftarrow \Sigma.\mathsf{Enc}(pk_{o,2}, m; r_2)\},$$

along with the label t^* and the witness (m_0, r_1, r_2). It receives back $(\mathsf{crs}', \pi'_{t^*})$, sets the main public key to $mpk := (pk_{o,1}, pk_{o,2}, pk_{u,1}, pk_{u,2}, \mathsf{crs} := \mathsf{crs}')$ and the challenge ciphertext to $C_{t^*} := (e_1^*, e_2^*, \pi_{t^*} := \pi'_{t^*})$. The adversary A makes Q_k function secret key queries (i.e., KeyGen'). All queries for a function f and tag t are answered by using the generated pair $(sk_{o,1}, sk_{u,1})$ and constructing PKey:1$[f, t, sk_{o,1}, sk_{u,1}, \mathsf{crs}]$. Similarly, all Q_t token queries for a pair of tags (t, t') are answered by using $sk_{o,1}$ and constructing PUpdate:1$[t, t', sk_{o,1}, mpk]$. If the query is to $\mathsf{HonTokGen}$, then B stores PUpdate:1$[t, t', sk_{o,1}, mpk]$ in $\mathcal{HT}$, otherwise (in case of $\mathsf{CorTokGen}$ query) B return PUpdate:1$[t, t', sk_{o,1}, mpk]$ to A. Enc' queries are simply answered using the previously constructed mpk. Lastly, $\mathsf{HonUpdate}$ queries for a tuple $(t, t', \cdot, \cdot)$ are answered using the program PUpdate:1$[t, t', sk_{o,1}, mpk]$, where it either already exists in $\mathcal{HT}$ or can be generated by B using $sk_{o,1}$.

If the zero-knowledge challenger of Π used the honest setup algorithm and the prover to generate crs' and π'_{t^*}, then we are exactly in hybrid Hybrid_0, and if it has used the simulator, then we are in hybrid Hybrid_1. Therefore, if A can distinguish the two hybrids with non-negligible advantage, then B can break the zero-knowledge property of Π.

Lemma 2. *If PKE scheme Σ is* $\mathsf{IND\text{-}CPA}$ *secure, then* $\mathsf{Hybrid}_1 \approx \mathsf{Hybrid}_2$.

Proof. We construct an adversary B for the $\mathsf{IND\text{-}CPA}$ security of Σ. First, B is given a target tag t^* and a pair of messages m_0, m_1 by A. Next, B generates the pairs of keys $(sk_{o,1}, pk_{o,1})$, $(sk_{u,1}, pk_{u,1})$, $(sk_{u,2}, pk_{u,2})$ and computes the encryption $e_1^* \leftarrow \Sigma.\mathsf{Enc}(pk_{o,1}, m_0; r_1)$. Then, B receives a public key pk' from the $\mathsf{IND\text{-}CPA}$ challenger of Σ and sets $pk_{o,2} := pk'$. Next, B submits the messages m_0, m_1 to the challenger and receives back e'. It sets $e_2^* = e'$ and uses the simulation algorithm to obtain

$$(\mathsf{crs}, \pi_{t^*}) \leftarrow \Pi.\mathsf{Sim}(1^\lambda, t^*, \{\exists m, r_1, r_2 \mid e_1^* \leftarrow \Sigma.\mathsf{Enc}(pk_{o,1}, m; r_1) \\ \wedge e_2^* \leftarrow \Sigma.\mathsf{Enc}(pk_{o,2}, m; r_2)\}),$$

with the tag t^* as the label. The main public key and the challenge ciphertext are set as $mpk := (pk_{o,1}, pk_{o,2}, pk_{u,1}, pk_{u,2}, \mathsf{crs})$ and $C_{t^*} := (e_1^*, e_2^*, \pi_{t^*})$. As in the proof of Lemma 1, B uses the secret keys $(sk_{o,1}, sk_{u,1})$ and the main public key mpk to answer all the queries.

If the IND-CPA challenger of Σ gave an encryption of m_0, then we are exactly in hybrid Hybrid_1, and if it gave an encryption of m_1, then we are in hybrid Hybrid_2. Therefore, if A can distinguish the two hybrids with non-negligible advantage, then B can break the IND-CPA security of Σ.

Lemma 3. *If $i\mathcal{O}$ is an indistinguishability obfuscator for the circuit class $\mathcal{C}_\lambda$, then it holds that* $\mathsf{Hybrid}_{3,i} \approx \mathsf{Hybrid}_{3,i+1}$ *for* $i \in [0, Q_k - 1]$.

Proof. We construct a distinguisher B for $i\mathcal{O}$. First, B is given a target tag t^* and a pair of messages m_0, m_1 by A. Next, B generates all public/secret keys that form mpk and msk, and computes the encryptions $e_1^* \leftarrow \Sigma.\mathsf{Enc}(pk_{o,1}, m_0; r_1)$ and $e_2^* \leftarrow \Sigma.\mathsf{Enc}(pk_{o,2}, m_1; r_2)$. Then, B uses the simulation algorithm to obtain

$$(\mathsf{crs}, \pi_{t^*}) \leftarrow \Pi.\mathsf{Sim}(1^\lambda, t^*, \{\exists m, r_1, r_2 \mid e_1^* \leftarrow \Sigma.\mathsf{Enc}(pk_{o,1}, m; r_1) \\ \wedge\, e_2^* \leftarrow \Sigma.\mathsf{Enc}(pk_{o,2}, m; r_2)\}),$$

with the tag t^* as the label. The main public key and the challenge ciphertext are set as $mpk := (pk_{o,1}, pk_{o,2}, pk_{u,1}, pk_{u,2}, \mathsf{crs})$ and $C_{t^*} := (e_1^*, e_2^*, \pi_{t^*})$. HonUpdate, HonTokGen, CorTokGen and Enc′ queries are answered in an analogous way to the proof of Lemma 1. For the function secret key queries, A makes Q_k queries to KeyGen′. For $j \leq i$, the j-th function secret key is created as an obfuscation of the program PKey:2$[f_j, t, sk_{o,2}, sk_{u,2}, \mathsf{crs}]$ for a tag t. For $j > i+1$, the j-th function secret key is created as an obfuscation of the program PKey:1$[f_j, t, sk_{o,1}, sk_{u,1}, \mathsf{crs}]$ for a tag t. For the $(i+1)$-th function secret key query B submits $C_0 =$ PKey:1$[f_{i+1}, t, sk_{o,1}, sk_{u,1}, \mathsf{crs}]$ and $C_1 =$ PKey:2$[f_{i+1}, t, sk_{o,2}, sk_{u,2}, \mathsf{crs}]$ to the $i\mathcal{O}$ challenger. It receives back an obfuscated circuit C', which B sets as the $(i+1)$-th function secret key.

Next, we show that both programs have the same input/output behavior. In case the inputs (e_1, e_2, π) consist of valid encryptions e_1, e_2 of the same message and π is a valid proof, then both programs decrypt to the same message m irrespective of the key used and compute the same function f_{i+1}. Hence, the output is the same on all inputs of this class. The second set of inputs we consider are when the proof π does not pass the verification (the second step in the programs). Then, both programs output $\perp$. Lastly, we consider class of inputs where π passes the verification, but the ciphertexts e_1, e_2 are not valid encryption of the same message. Due to the (one-time) statistical simulation-soundness property of the ℓ-SSS-NIZK, this can only happen if $e_1 = e_1^*$ and $e_2 = e_2^*$. In this case decrypting e_1^* gives m_0 and decrypting e_2^* gives m_1. Though, due to the validity of the IND-CUFE-CPA adversary as defined in Section 3, the output of the first program $f_{i+1}(m_0)$ and the output of the second program $f_{i+1}(m_1)$ are equal. This concludes that both programs have the same output on all inputs.

If the $i\mathcal{O}$ challenger chose the first program, then we are exactly in hybrid $\mathsf{Hybrid}_{3,i}$, and if it chose the second program, then we are in hybrid $\mathsf{Hybrid}_{3,i+1}$. Therefore, if A can distinguish the two hybrids with non-negligible advantage, then B can break the security of $i\mathcal{O}$ for the circuit class $\mathcal{C}_\lambda$.

Lemma 4. *If $pi\mathcal{O}$ is a probabilistic indistinguishability obfuscator for the sampler $\mathcal{S}^{\Sigma,\Pi}$, then it holds that* $\mathsf{Hybrid}_{4,i} \approx \mathsf{Hybrid}_{4,i+1}$ *for* $i \in [0, Q_t - 1]$.

Proof. We construct a distinguisher B for $pi\mathcal{O}$. First, B generates all key pairs $\mathsf{K} := ((sk_{o,1}, pk_{o,1}), (sk_{o,2}, pk_{o,2}), (sk_{u,1}, pk_{u,1}), (sk_{u,2}, pk_{u,2}))$, sets $mpk := (pk_{o,1}, pk_{o,2}, pk_{u,1}, pk_{u,2}, \mathsf{crs})$ and challenge ciphertext $C_{t^*} := (e_1^*, e_2^*, \pi_{t^*})$ using the generated keys as in the proof of Lemma 3. Moreover, B uses the secret keys $(sk_{o,2}, sk_{u,2})$ and the main public key mpk to answers KeyGen′ and Enc′ oracle queries, respectively. For the Q_t token queries that A makes B proceeds as follows. For $j \leq i$, the j-th token is created as an obfuscation of the program PUpdate:2$[t_j, t'_j, sk_{o,2}, mpk]$. For $j > i+1$, the j-th token is created as an obfuscation of the program PUpdate:1$[t_j, t'_j, sk_{o,1}, mpk]$. If the query is to HonTokGen, then B stores the corresponding program in $\mathcal{HT}$, otherwise (in case of CorTokGen query) B return the program to A. Similarly, HonUpdate queries are answered either by using an already stored program in $\mathcal{HT}$ or by generating a new program using $sk_{o,2}$. For the $(i+1)$-th token query (for the pair $\mathsf{T} := (t_{i+1}, t'_{i+1})$), B uses the challenger of $pi\mathcal{O}$, which samples $(C_0, C_1, z) \leftarrow D^{\mathsf{T},\mathsf{K}}$, where $C_0 =$ PUpdate:1$[t_{i+1}, t'_{i+1}, sk_{o,1}, mpk]$, $C_1 =$ PUpdate:2$[t_{i+1}, t'_{i+1}, sk_{o,2}, mpk]$ and $z = (t_{i+1}, t'_{i+1}, mpk)$. The challenger generates C' (obfuscation of either C_0 or C_1) and gives C' to B, which B sets as the $(i+1)$-th token.

If the $pi\mathcal{O}$ challenger chose the first program, then we are exactly in hybrid $\mathsf{Hybrid}_{4,i}$, and if it chose the second program, then we are in hybrid $\mathsf{Hybrid}_{4,i+1}$. Therefore, if A can distinguish the two hybrids with non-negligible advantage, then B can break the security of $pi\mathcal{O}$ for the sampler $\mathcal{S}^{\Sigma,\Pi}$.

Lemma 5. *If PKE scheme Σ is* IND-CPA *secure, then* $\mathsf{Hybrid}_{4,Q_t} \approx \mathsf{Hybrid}_5$.

Proof. The proof of this lemma follows analogously to that of Lemma 2.

Lemma 6. *If $pi\mathcal{O}$ is a probabilistic indistinguishability obfuscator for the sampler $\mathcal{S}^{\Sigma,\Pi}$, then it holds that* $\mathsf{Hybrid}_{6,i} \approx \mathsf{Hybrid}_{6,i+1}$ *for* $i \in [0, Q_t - 1]$.

Proof. The proof of this lemma follows analogously to that of Lemma 4.

Lemma 7. *If $i\mathcal{O}$ is an indistinguishability obfuscator for the circuit class $\mathcal{C}_\lambda$, then it holds that* $\mathsf{Hybrid}_{7,i} \approx \mathsf{Hybrid}_{7,i+i}$ *for* $i \in [0, Q_k - 1]$.

Proof. The proof of this lemma follows analogously to that of Lemma 3.

Lemma 8. *If ℓ-SSS-NIZK proof system Π is computationally zero-knowledge, then it holds that* $\mathsf{Hybrid}_{7,Q_k} \approx \mathsf{Hybrid}_8$.

Proof. The proof of this lemma follows analogously to that of Lemma 1.

This concludes the proof of Theorem 1. □

Instantiations of the Generic Construction Analogous to the universal proxy re-encryption construction of Döttling and Nishimaki [DN21], the class of PKE that we can use depends on the security level of piO. Since we do not want to impose any restrictions on the underlying PKE scheme and to allow instantiating our generic construction using any IND-CPA PKE scheme, we only consider piO with stronger security, namely, dynamic-input piO, and the recent work of doubly-probabilistic iO [ACH20].

Instantiation by dynamic-input probabilistic iO. Using a dynamic-input piO we can relax the requirements of the underlying encryption scheme and use any IND-CPA secure PKE scheme. We note that dynamic-input piO is a generalization of differing-input iO by Garg et al. [GGHW14] to randomized circuits, hence, it inherits the implausibility results of differing-input iO. This implies that dynamic-input piO for general dynamic-input indistinguishable samplers is implausible. However, Canetti et al. [CLTV15] argued that a construction of dynamic-input piO for specific classes of samplers is possible, and here we conjecture that dynamic-input piO for class of samplers $\mathcal{S}^{\Sigma,\Pi}$ might exist.

Conjecture 1. A dynamic-input piO for the class of samplers $\mathcal{S}^{\Sigma,\Pi}$ exists.

We note that our conjecture is quite similar to a standard (and believed to hold) conjecture used in previous works, such as [CLTV15] and [DN21].

Corollary 1. *If there exists dynamic-input piO for the class of samplers $\mathcal{S}^{\Sigma,\Pi}$, where Σ is an* IND-CPA *PKE scheme and Π is an ℓ-SSS-NIZK proof system, then our generic construction is selectively* IND-CUFE-CPA *secure CUFE scheme for any* IND-CPA *encryption scheme Σ.*

Instantiation by doubly-probabilistic iO. Following the recent work of Agrikola et al. [ACH20], if we assume the exponential DDH assumption and polynomially secure iO, then we can instantiate our construction using any IND-CPA PKE scheme.

Theorem 2. *If there exists polynomially secure iO and the exponential DDH assumption holds, then our generic construction is selectively* IND-CUFE-CPA *secure CUFE scheme for any* IND-CPA *PKE Σ.*

4.2 Extending Supported Predicates

For our generic construction it is easily possible to extend it from supporting the equality test predicate (i.e., tags) to more powerful predicates, i.e., an access control mechanism known from ABE in the terminology of [ACGU20].

Let us follow the notation of Gorbunov et al. [GVW13], who construct ABE for any circuit of arbitrary polynomial size. Thus, let ind be an ℓ bit public index (used for encryption) and $\mathbf{P}$ a Boolean predicate (associated to secret keys) and decryption should only work if $\mathbf{P}(\mathsf{ind}) = 1$. Now, we can simply associate function keys with more expressive predicates $\mathbf{P}$ (encode them into PKey) instead of tags and use as public labels for the NIZK the public index ind (i.e., the attributes). In the decryption circuit $P_{f,\mathbf{P}}$, one simply checks if for label ind and hard-coded $\mathbf{P}$ it holds that $\mathbf{P}(\mathsf{ind}) = 1$.

Switching the public index in a ciphertext from ind to some ind', i.e., change the attributes in the ciphertext, can simply be done by viewing the public indices as the tags in the current solution. Now this represents a generalization of our generic construction where we only have the equality predicate $\mathbf{P}_t(\hat{t}) = 1$ if and only if $t = \hat{t}$.

5 Lattice-Based CUFE Construction for Inner Products

After recalling the syntax and properties of the main sampling algorithms used in lattice-based constructions, we will build a CUFE scheme for inner-products from the LWE assumption in the random oracle model in this section. For a further exposition of lattice preliminaries we refer the reader to Appendix A.4.

5.1 Lattice Definitions and Algorithms

For any matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, we define the orthogonal q-ary lattice of $\mathbf{A}$ as $\Lambda_q^{\perp}(\mathbf{A}) := \{\vec{u} \in \mathbb{Z}^m : \mathbf{A}\vec{u} = \vec{0} \mod q\}$.

The normal Gaussian distribution of mean 0 and variance σ^2 is the distribution on $\mathbb{R}$ with probability density function $\frac{1}{\sigma\sqrt{2\pi}}\frac{1}{e^{x^2/(2\sigma^2)}}$. The lattice Gaussian distribution with support a lattice $\Lambda \subseteq \mathbb{Z}^m$, standard deviation σ and centered at $\vec{c} \in \mathbb{Z}^m$, is defined as:

$$\text{for all } \vec{y} \in \Lambda : \mathcal{D}_{\Lambda,\sigma,\vec{c}}(\vec{y}) = \frac{e^{-\pi\|\vec{y}-\vec{c}\|^2/\sigma^2}}{\sum_{\vec{x}\in\Lambda} e^{-\pi\|\vec{x}-\vec{c}\|^2/\sigma^2}}$$

The following algorithms will be used in lattice construction, and their properties needed in the security proof.

Lemma 9 ([GPV08] Preimage Sampable Functions). *For any prime $q = poly(n)$, any $m \geq 5n\log q$, and any $s \geq m^{2.5}\omega(\sqrt{\log m})$, it holds that there exist PPT algorithms* $\mathsf{TrapGen}$, $\mathsf{SampleD}$, $\mathsf{SamplePre}$ *such that:*

1. $\mathsf{TrapGen}$ *computes $(\mathbf{A}, \mathbf{T}) \leftarrow \mathsf{TrapGen}(1^n, 1^m)$, where $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ is statistically close to uniform and $\mathbf{T} \subset \Lambda_q^{\perp}(\mathbf{A})$ is a basis with $\|\widetilde{\mathbf{T}}\| \leq m^{2.5}$. The matrix $\mathbf{A}$ (and q) is public, while the good basis $\mathbf{T}$ is the trapdoor.*
2. $\mathsf{SampleD}$ *samples matrices $\mathbf{Z}'$ from $\mathcal{D}_{\mathbb{Z}^{m\times m},s}$,*
3. *The trapdoor inversion algorithm* $\mathsf{SamplePre}(\mathbf{A}, \mathbf{T}, \mathbf{D}, s)$, *for $\mathbf{D} \in \mathbb{Z}_q^{n\times m}$, outputs a matrix $\mathbf{Z} \in \mathbb{Z}^{m\times m}$ such that $\mathbf{AZ} = \mathbf{D}$.*

In addition, it holds that the following distributions D_1, D_2 are statistically close:

$$D_1 = (\mathbf{A}, \mathbf{Z}, \mathbf{D}),\ \textit{s.t.}\ (\mathbf{A}, \mathbf{T}) \leftarrow \mathsf{TrapGen}(1^n, 1^m), \mathbf{D} \leftarrow \mathbb{Z}_q^{n\times m},$$
$$\mathbf{Z} \leftarrow \mathsf{SamplePre}(\mathbf{A}, \mathbf{T}, \mathbf{D}, s),$$

$$D_2 = (\mathbf{A}, \mathbf{Z}', \mathbf{AZ}'),\ \textit{where}\ \mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}, \mathbf{Z}' \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m},s}.$$

Theorem 3 ([ABB10] SampleLeft). *Let $q > 2$, full rank $\mathbf{A}, \mathbf{B} \in \mathbb{Z}_q^{n\times m}$ with $m > n$, a basis $\mathbf{T_A}$ of $\Lambda_q^{\perp}(\mathbf{A})$, a matrix $\mathbf{D} \in \mathbb{Z}_q^{n\times m}$ and $\sigma > \|\widetilde{\mathbf{T}}_{\mathbf{A}}\| \cdot \omega(\sqrt{\log m})$. Then there exists PPT algorithm* $\mathsf{SampleLeft}(\mathbf{A}, \mathbf{T_A}, \mathbf{B}, \mathbf{D}, \sigma)$ *that outputs a matrix $\mathbf{X} \in \mathbb{Z}^{2m\times m}$, distributed statistically close to $\mathcal{D}_{\Lambda_q^{\mathbf{D}}(\mathbf{A}|\mathbf{B}),\sigma}$.*

5.2 Lattice Construction

We are building on the work of Abdalla et al. [ACGU20], who gave the first constructions, one in the standard model (SM) and one in the random oracle (RO) model, of a lattice-based identity-based IPFE scheme, and proved their security[9] under the $\mathsf{LWE}_{q,\alpha,n}$ assumption (Definition 10). Their constructions are in turn based on the IPFE scheme of Agrawal et al. [ALS16], ALS, described in Figure 4.

$\underline{\mathsf{Setup}(1^\lambda, n)}$: $\mathbf{A} \leftarrow_\$ \mathbb{Z}_{q_{\mathsf{ALS}}}^{n\times m}$ $\mathbf{Z} \leftarrow_\$ D_{\mathbb{Z}^{m\times \ell},\rho_{\mathsf{ALS}}}$ $\mathbf{D} \leftarrow \mathbf{AZ}$ $\mathsf{mpk} \leftarrow (\mathbf{A}, \mathbf{D})$ $\mathsf{msk} \leftarrow \mathbf{Z}$ Return $(\mathsf{mpk}, \mathsf{msk})$ $\underline{\mathsf{KeyGen}(\mathbf{Z}, \vec{y} \in \mathcal{Y})}$: Return $(\vec{y}, \mathsf{sk}_{\vec{y}} := \mathbf{Z}\cdot\vec{y})$	$\underline{\mathsf{Enc}(\mathsf{mpk}, \vec{x} \in \mathcal{X})}$: $\vec{s} \leftarrow_\$ \mathbb{Z}_{q_{\mathsf{ALS}}}^{n}$ $\vec{e}_1 \leftarrow_\$ \mathcal{D}_{\mathbb{Z}^m,\sigma_{\mathsf{ALS}}}$ $\vec{e}_2 \leftarrow_\$ \mathcal{D}_{\mathbb{Z}^\ell,\sigma_{\mathsf{ALS}}}$ $\mathsf{ct}_1 = \mathbf{A}^\top\vec{s} + \vec{e}_1$ $\mathsf{ct}_2 = \mathbf{D}^\top\vec{s} + \vec{e}_2 + \lfloor \frac{q}{K} \rfloor \cdot \vec{x}$ Return $(\mathsf{ct}_1, \mathsf{ct}_2)$ $\underline{\mathsf{Dec}(\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{sk}_{\vec{y}}, \vec{y} \in \mathcal{Y})}$: $\mu' = \vec{y}^\top \cdot \mathsf{ct}_2 - \mathsf{sk}_{\vec{y}}^\top \cdot \mathsf{ct}_1$ Return $\arg\min_{\mu\in\{0,\dots,K+1\}} \left\lvert \lfloor \frac{q}{K} \rfloor \cdot \mu' - \mu \right\rvert$

Fig. 4. Inner-product functional encryption scheme ALS, with parameters as in [ACGU20].

In our construction, we start from the RO scheme of Abdalla et al. [ACGU20] and enhanced their design in order to allow distinguishing fresh and updated ciphertexts. To prove its security, we rely on the programmability of random oracles $H_1, H_2, H_3 \colon \mathcal{T} \to \mathbb{Z}_q^{n\times m}$, where $\mathcal{T}$ is the tag-space. Notice that programmability of random oracles is required in the security proof to simulate the new supported functionality, i.e., updating ciphertexts. Thus, even though our construction is only proved secure in the RO model, it also supports a richer class of functionalities than previous works. Our lattice-based CUFE construction is described in Figure 5. Dimensions of matrices involved in the construction are presented in Table 1.

Table 1. Matrices, vectors, and respective dimensions used in the construction.

$\mathbf{A}$	$\mathbb{Z}_q^{n\times m}$	$\mathbf{X}_{t,t'}$	$\mathbb{Z}^{m\times m}$
$\mathbf{T_A}$	$\mathbb{Z}^{m\times m}$	$\mathbf{Y}_{t,t'}$	$\mathbb{Z}^{m\times m}$
$\mathbf{B}_{t,1}$	$\mathbb{Z}_q^{n\times m}$	$\vec{s}$	$\mathbb{Z}_q^{n}$
$\mathbf{B}_{t,2}$	$\mathbb{Z}_q^{n\times m}$	$\vec{e}_1$	$\mathbb{Z}^{m}$
$\mathbf{D}_t$	$\mathbb{Z}_q^{n\times m}$	$\vec{e}_2$	$\mathbb{Z}^{m}$
$\Delta_{t\to t',1}$	$\mathbb{Z}^{2m\times 2m}$	$\vec{e}_3$	$\mathbb{Z}^{m}$
$\Delta_{t\to t',2}$	$\mathbb{Z}^{2m\times m}$	$\mathbf{S}$	$\{\pm 1\}^{m\times m}$
$\mathsf{ct}_{t,1,1}$	$\mathbb{Z}_q^{2m}$	$\vec{f}_1$	$\mathbb{Z}^{2m}$
$\mathsf{ct}_{t,1,2}$	$\mathbb{Z}_q^{m}$	$\vec{f}_2$	$\mathbb{Z}^{m}$
$\mathsf{ct}_{t,2,1}$	$\mathbb{Z}_q^{2m}$	$\vec{f}$	$\mathbb{Z}^{m}$
$\mathsf{ct}_{t,2,2}$	$\mathbb{Z}_q^{m}$	$\vec{x}$	$\{0,\dots,P\}^m$
$\mathbf{Z}_{t,1}$	$\mathbb{Z}^{2m\times m}$	$\vec{y}$	$\{0,\dots,V\}^m$
$\mathbf{Z}_{t,2}$	$\mathbb{Z}^{2m\times m}$	$\langle \vec{y}, \vec{x}\rangle$	$\{0,\dots,mPV\}$

The first component of the ciphertext, $\mathsf{ct}_{t,1,1}$, depends on the tag t but not on the message. The second component, $\mathsf{ct}_{t,1,2}$, on the other hand, depends on the message $\vec{x}$ to be encrypted. The two components are intertwined by the shared randomness $\vec{s} \in \mathbb{Z}_q^n$. In order to update ciphertexts, it is therefore necessary to update the two parts of a given ciphertext to the prescribed new tag, while preserving the common randomness, the underlying plaintext, and, at the same time, without increasing the error term too much. Latter would prevent correct decryption of updated ciphertexts.

[9] We refer the reader to Appendix A.3 for a formal definition.

$\underline{\mathsf{Setup}(1^\lambda, n)\text{:}}$
$(\mathbf{A}, \mathbf{T_A}) \leftarrow \mathsf{TrapGen}(1^n, 1^m)$
Return $(\mathsf{mpk} := \mathbf{A}, \mathsf{msk} := (\mathbf{A}, \mathbf{T_A}))$

$\underline{\mathsf{KeyGen}((\mathbf{A}, \mathbf{T_A}), \vec{y} \in \mathcal{Y}, t)\text{:}}$
for $\ell = 1, 2 : \mathbf{Z}_{t,\ell} \leftarrow \mathsf{SampleLeft}(\mathbf{A}, \mathbf{T_A}, H_\ell(t), H_3(t), \rho_\ell)$
Return $(\vec{y}, \{\mathsf{sk}_{\vec{y},t,\ell} := \mathbf{Z}_{t,\ell} \cdot \vec{y}\}_{\ell=1,2})$

$\underline{\mathsf{TokGen}((\mathbf{A}, \mathbf{T_A}), t, t')\text{:}}$
$\mathbf{B}_{t,1} := H_1(t)$, $\mathbf{B}_{t',2} := H_2(t')$, $\mathbf{D}_t := H_3(t)$, $\mathbf{D}_{t'} := H_3(t')$, $\mathbf{Y}_{t,t'} \leftarrow_\$ \mathcal{D}_{\mathbb{Z}^{m\times m},\rho}$
$\mathbf{X}_{t,t'} \leftarrow \mathsf{SamplePre}(\mathbf{A}, \mathbf{T_A}, \mathbf{B}_{t',2} - \mathbf{B}_{t,1}\mathbf{Y}_{t,t'}, \rho)$
$\Delta_{t\to t',1} := \left[\begin{array}{c|c} \mathbf{I}_m & \mathbf{X}_{t,t'} \\ \hline \mathbf{0} & \mathbf{Y}_{t,t'} \end{array}\right]$
$\Delta_{t\to t',2} \leftarrow \mathsf{SampleLeft}(\mathbf{A}, \mathbf{T_A}, \mathbf{B}_{t,1}, \mathbf{D}_{t'} - \mathbf{D}_t, \rho)$
Return $(\Delta_{t\to t',1}, \Delta_{t\to t',2})$

$\underline{\mathsf{Enc}(\mathsf{mpk}, \vec{x} \in \mathcal{X}, t)\text{:}}$
$\mathbf{B}_{t,1} := H_1(t)$, $\mathbf{D}_t := H_3(t)$
$\vec{s} \leftarrow_\$ \mathbb{Z}_q^n$, $\vec{e}_1, \vec{e}_2 \leftarrow_\$ \mathcal{D}_{\mathbb{Z}^m,\sigma}$, $\vec{e}_3 \leftarrow_\$ \mathcal{D}_{\mathbb{Z}^m,\mu}$, $\mathbf{S} \leftarrow_\$ \{\pm 1\}^{m\times m}$
$\mathsf{ct}_{t,1,1} := \mathbf{H}_{t,1}^\top \vec{s} + \vec{f}$ with $\mathbf{H}_{t,1} := (\mathbf{A}|\mathbf{B}_{t,1})$, $\vec{f} := (\mathbf{I}_m|\mathbf{S})^\top \cdot \vec{e}_1$
$\mathsf{ct}_{t,1,2} := \mathbf{D}_t^\top \vec{s} + \vec{e}_2 + \vec{e}_3 + \lfloor \frac{q}{K} \rfloor \cdot \vec{x}$
Return $(\mathsf{ct}_{t,1,1}, \mathsf{ct}_{t,1,2})$

$\underline{\mathsf{Update}(\Delta_{t\to t',1}, \Delta_{t\to t',2}, \mathsf{ct}_{t,1,1}, \mathsf{ct}_{t,1,2})\text{:}}$
$\mathbf{B}_{t',2} := H_2(t')$, $\mathbf{D}_{t'} := H_3(t')$, $\vec{r} \leftarrow_\$ \mathbb{Z}_q^n$, $\vec{f}_1 \leftarrow_\$ \mathcal{D}_{\mathbb{Z}^{2m},\tau}$, $\vec{f}_2 \leftarrow_\$ \mathcal{D}_{\mathbb{Z}^m,\tau}$
$\mathsf{ct}_{t',2,1} := \Delta_{t\to t',1}^\top \mathsf{ct}_{t,1,1} + \mathbf{H}_{t',2}^\top \vec{r} + \vec{f}_1$ with $\mathbf{H}_{t',2} = (\mathbf{A}|\mathbf{B}_{t',2})$
$\mathsf{ct}_{t',2,2} := \mathsf{ct}_{t,1,2} + \Delta_{t\to t',2}^\top \mathsf{ct}_{t,1,1} + \mathbf{D}_{t'}^\top \vec{r} + \vec{f}_2$
Return $(\mathsf{ct}_{t',2,1}, \mathsf{ct}_{t',2,2})$

$\underline{\mathsf{Dec}(\mathsf{ct}_{t,\ell,1}, \mathsf{ct}_{t,\ell,2}, \vec{y}, \{\mathsf{sk}_{\vec{y},t,\ell}\}_{\ell=1,2})\text{:}}$
$\mu' \leftarrow \vec{y}^\top \cdot \mathsf{ct}_{t,\ell,2} - \mathsf{sk}_{\vec{y},t,\ell}^\top \cdot \mathsf{ct}_{t,\ell,1}$
Return $\arg\min_{\mu \in \{0,\ldots,K+1\}} \left| \lfloor \frac{q}{K} \rfloor \cdot \mu - \mu' \right|$

Fig. 5. Lattice-based Ciphertext-Updatable IPFE scheme.

This can be done using techniques inspired by [FL19, CCL+14]. Moreover, since the randomness is given by uniform vector in $\mathbb{Z}_q^n$ and the encryption scheme is additively homomorphic, ciphertexts can be easily re-randomized.

To update a ciphertext from t to t', we want to produce a $2m \times 2m$ matrix $\Delta_{t\to t',1}$ over $\mathbb{Z}$ and a $2m \times m$ matrix $\Delta_{t\to t',2}$ over $\mathbb{Z}$, with $\Delta_{t\to t',2} \leftarrow \mathcal{D}_{\mathbb{Z}^{2m\times m},\rho}$. $\Delta_{t\to t',1}$ has the form

$$\Delta_{t\to t',1} := \left[\begin{array}{c|c} \mathbf{I}_m & \mathbf{X}_{t,t'} \\ \hline \mathbf{0} & \mathbf{Y}_{t,t'} \end{array}\right],$$

with $\mathbf{X}_{t,t'}, \mathbf{Y}_{t,t'} \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m},\rho}$. $\Delta_{t\to t',1}$ and $\Delta_{t\to t',2}$ are additionally conditioned on

$$\mathbf{H}_{t,1} \cdot \Delta_{t\to t',1} = \mathbf{H}_{t',2}, \qquad \text{and} \qquad \mathbf{H}_{t,1} \cdot \Delta_{t\to t',2} = \mathbf{D}_{t'} - \mathbf{D}_t.$$

In the real game the matrix $\Delta_{t\to t',1}$ and $\Delta_{t\to t',2}$ will be produced using the trapdoor $\mathbf{T_A}$, i.e., $\mathbf{Y}_{t,t'}$ will be sampled from $\mathcal{D}_{\mathbb{Z}^{m\times m},\rho}$, $\mathbf{X}_{t,t'}$ using $\mathsf{SamplePre}(\mathbf{A}, \mathbf{T_A}, \mathbf{B}_{t',2} - \mathbf{B}_{t,1}\mathbf{Y}_{t,t'}, \rho)$, where $\mathbf{H}_{t',2} = (\mathbf{A}|\mathbf{B}_{t',2})$, and $\Delta_{t\to t',2}$ using $\mathsf{SampleLeft}(\mathbf{A}, \mathbf{T_A}, \mathbf{B}_{t,1}, \mathbf{D}_{t'} - \mathbf{D}_t, \rho)$.

Vice versa, in the security proof, we will leverage on the programmability of the random oracles H_1, H_2, and H_3: whenever the source tag t equals the challenge tag t^*, $\mathbf{X}_{t,t'}$, $\mathbf{Y}_{t,t'}$, and $\Delta_{t\to t',2}$ will be sampled from the appropriate distributions, $H_2(t') = \mathbf{B}_{t',2}$ will be set to equal $\mathbf{A}\mathbf{X}_{t,t'} + \mathbf{B}_{t,1}\mathbf{Y}_{t,t'}$, and $H_3(t') = \mathbf{D}_{t'}$ to $\mathbf{H}_{t,1} \cdot \Delta_{t\to t',2} + \mathbf{D}_t$. For all other pair of tags, t, t', the token $(\Delta_{t\to t',1}, \Delta_{t\to t',2})$ is produced using the trapdoor of $H_1(t) = \mathbf{B}_{t,1}$: the matrix $\mathbf{B}_{t,1}$ will be produced using the $\mathsf{TrapGen}$ algorithm, and the update token will be produced using such trapdoor.

To update a ciphertext $(\mathsf{ct}_{t,1,1}, \mathsf{ct}_{t,2,2})$, given the appropriate token $(\Delta_{t\to t',1}, \Delta_{t\to t',2})$, fresh randomness $\vec{r} \leftarrow \mathbb{Z}_q^n$ and noises $\vec{f}_1 \leftarrow \mathcal{D}_{\mathbb{Z}^{2m},\tau}, \vec{f}_2 \leftarrow \mathcal{D}_{\mathbb{Z}^m,\tau}$ are sampled and the new ciphertext

$(\mathsf{ct}_{t',2,1}, \mathsf{ct}_{t',2,2})$ is computed as

$$\mathsf{ct}_{t',2,1} := \Delta_{t\to t',1}^\top \mathsf{ct}_{t,1,1} + \mathbf{H}_{t',2}^\top \vec{r} + \vec{f}_1,$$
$$\mathsf{ct}_{t',2,2} := \mathsf{ct}_{t,1,2} + \Delta_{t\to t',2}^\top \mathsf{ct}_{t,1,1} + \mathbf{D}_{t'}^\top \vec{r} + \vec{f}_2.$$

The functional secret keys, $\{sk_{t,\ell,\vec{y}}\}_{\ell=1,2}$, can be produced as follows:

- for the challenge tag t^*: for $\ell = 1$, using the ALS challenger, and for $\ell = 2$, using the trapdoor of $\mathbf{B}_{t^*,2}$.
- for tags, $t \neq t^*$, for which no update token of the form $(\Delta_{t^*\to t,1}, \Delta_{t^*\to t,2})$ was queried but to which the challenge ciphertext was updated: using the trapdoor of $\mathbf{B}_{t,1}$, or again the ALS challenger for $\ell = 2$.
- for all other tags: using the trapdoor of $\mathbf{B}_{t,\ell}$ for $\ell = 1, 2$.

Parameters and Correctness. In our construction, ciphertexts encode vectors $\vec{x} \in \{0, \dots, P\}^m$ under a tag t. Secret keys corresponds to a tag t and a vector $\vec{y} \in \{0, \dots, V\}^m$. When tags match, our scheme decrypts the bounded inner-product $\langle \vec{x}, \vec{y} \rangle \in \{0, \dots, K\}$, where $K = mPV$. Moreover, our scheme parameters must satisfy the following bounds:

- $m \geq 6n \log q$ (required by $\mathsf{TrapGen}$),
- $\alpha q > 2\sqrt{n}$ (required by hardness of LWE).
- $\rho = \rho_1 = \rho_{\mathrm{ALS}} \geq m^{2.5} \cdot \omega(\sqrt{\log m})$ (required by $\mathsf{SamplePre}$),
- $\rho_2 \geq m\rho \cdot \lambda^{\omega(1)}$ (required in the security proof for the indistinguishability of function keys),
- $\sigma = \sigma_{\mathrm{ALS}}$,
- $\mathsf{NoiseGen}$: the spectral norm of $\mathbf{S}_{t^*}$ can be upper-bounded (by using the Frobenius norm) by m. Using Lemma 15, $s_1(\mathbf{Z}_{t^*}) \leq 3C\rho\sqrt{m}$, which implies $\mu \geq 3C\rho m^{1.5}$,
- $\tau \geq \sqrt{m}(\sigma + \mu + 2\sqrt{2}\rho\sigma m^{1.5} C')\lambda^{\omega(1)}$ (require in the security proof for the indistinguishability of updated honest ciphertexts) and $\tau \geq (\sigma\sqrt{m} + \sigma\rho_2 m^{1.5} + \sqrt{2}m^2\sigma\rho_2 C')\lambda^{\omega(1)}$ (for the indistinguishability of updates of the challenge ciphertext). Thus, we set $\tau \geq \max\{\sqrt{m}(\sigma + \mu + 2\sqrt{2}\rho\sigma m^{1.5}C'), (\sigma\sqrt{m} + \sigma\rho_2 m^{1.5} + \sqrt{2}m^2\sigma\rho_2 C')\} \cdot \lambda^{\omega(1)}$,
- $q > 2KVm(\sigma + \mu + \tau + 12\sqrt{2}C'm^{2.5}\rho_2(\rho\sigma + \tau))$ (required for successful decryption of updated ciphertexts),

Lemma 10 (Correctness). *For $q > 2KVm(\sigma + \mu + \tau + 12\sqrt{2}C'm^{2.5}\rho_2(\rho\sigma + \tau))$, the decryption of (updated) ciphertexts from the scheme in Fig. 5 is, w.h.p., correct.*

Proof. The correct decryption of fresh ciphertexts follows directly from the correctness of the Abdalla et al. [ACGU20] construction. On the other hand, an updated ciphertext has the following form:

$$\begin{aligned}
\mathsf{ct}_{t',2,1} &:= \Delta_{t\to t',1}^\top \mathsf{ct}_{t,1,1} + \mathbf{H}_{t',2}^\top \vec{r} + \vec{f}_1 \\
&= \mathbf{H}_{t',2}^\top (\vec{s} + \vec{r}) + \Delta_{t\to t',1}^\top \vec{f} + \vec{f}_1, \text{ and} \\
\mathsf{ct}_{t',2,2} &:= \mathsf{ct}_{t,1,2} + \Delta_{t\to t',2}^\top \mathsf{ct}_{t,1,1} + \mathbf{D}_{t'}^\top \vec{r} + \vec{f}_2 \\
&= \mathbf{D}_{t'}^\top (\vec{s} + \vec{r}) + \vec{e}_2 + \vec{e}_3 + \Delta_{t\to t',2}^\top \vec{f} + \vec{f}_2 + \left\lfloor \frac{q}{K} \right\rfloor \cdot \vec{x}.
\end{aligned}$$

Therefore, during decryption of updated ciphertexts, one obtains:

$$\begin{aligned}
\mu' &= \vec{y}^\top \cdot \mathsf{ct}_{t,2,2} - \mathsf{sk}_{t,2,\vec{y}}^\top \cdot \mathsf{ct}_{t,2,1} \\
&= \left\lfloor \frac{q}{K} \right\rfloor \langle \vec{y}, \vec{x} \rangle + \underbrace{\vec{y}^\top (\vec{e}_2 + \vec{e}_3 + \Delta_{t\to t',2}^\top \vec{f} + \vec{f}_2) - \vec{y}^\top \mathbf{Z}_{t',2}^\top (\Delta_{t\to t',1}^\top \vec{f} + \vec{f}_1)}_{\text{error terms}},
\end{aligned}$$

where we have used the fact that $\mathbf{H}_{t',2} \cdot \mathbf{Z}_{t',2} = \mathbf{D}_{t'}$. This decrypts correctly as long as the error terms obtained

$$\vec{y}^\top (\vec{e}_2 + \vec{e}_3 + \Delta_{t\to t',2}^\top \vec{f} + \vec{f}_2 - \mathbf{Z}_{t',2}^\top (\Delta_{t\to t',1}^\top \vec{f} + \vec{f}_1)),$$

are small compared to q/K. Since $\Delta_{t,t',1} \in \mathbb{Z}^{2m\times 2m}$, and $\Delta_{t,t',2}, \mathbf{Z}_{t',2} \in \mathbb{Z}^{2m\times m}$ are sampled via the $\mathsf{SamplePre}$ algorithm with parameter ρ and ρ_2 respectively, by Lemma 11, we know that $\|\mathbf{Z}_{t',2}\| \leq 2m \cdot \rho_2$, $\|\Delta_{t\to t',1}\| \leq 2m \cdot \rho$, and $\|\Delta_{t\to t',2}\| \leq \sqrt{2} \cdot m \cdot \rho$, as long as $\rho, \rho_2 \geq m^{2.5}\omega(\sqrt{\log n})$. Using again Lemma 11 and Lemma 14, we can also deduce that $\|\vec{e}_1\|, \|\vec{e}_2\| \leq \sigma\sqrt{m}$, $\|\vec{e}_3\| \leq \mu\sqrt{m}$, $\|\vec{f}\| \leq$

$C'\sigma\sqrt{2}m$ and $\|\vec{f_1}\| \leq \tau\sqrt{2m}, \|\vec{f_2}\| \leq \tau\sqrt{m}$, as long as $\sigma, \mu, \tau \geq \omega(\sqrt{\log n})$. Therefore, $\|\Delta_{t\to t',1}^\top \vec{f}\| \leq 2\sqrt{2}C'm^2\rho\sigma$, $\|\Delta_{t\to t',2}^\top \vec{f}\| \leq 2C'm^2\rho\sigma$, and $\|\mathbf{Z}_{t',2}^\top(\Delta_{t\to t',1}^\top \vec{f} + \vec{f_1})\| \leq 2m\rho_2(2\sqrt{2}C'm^2\rho\sigma + \sqrt{2m}\tau)$. Since, $\|\vec{y}\| \leq V\sqrt{m}$, the final error term is upper bounded by $V\sqrt{m} \cdot (\sigma\sqrt{m} + \mu\sqrt{m} + 2C'm^2\rho\sigma + \tau\sqrt{m} + 2m\rho_2(2\sqrt{2}C'm^2\rho\sigma + \sqrt{2m}\tau))$. For decryption to succeed, we want that the error term is smaller than $\frac{q}{2K}$, which implies:

$$\begin{aligned} q &> 2KV\sqrt{m} \cdot (\sigma\sqrt{m} + \mu\sqrt{m} + 2C'm^2\rho\sigma + \tau\sqrt{m} + 2m\rho_2(2\sqrt{2}C'm^2\rho\sigma + \sqrt{2m}\tau)) \\ &> 2KVm(\sigma + \mu + \tau + 12\sqrt{2}C'm^{2.5}\rho_2(\rho\sigma + \tau)). \end{aligned}$$ □

Security Proof. We now show that the adaptive security of our CUFE construction follows from the security of the ALS scheme. In order to do so, we however have to make the following restrictions regarding the validity of the adversary in the IND-CUFE-CPA experiment:

1. if $(\cdot, t^*, t') \in \mathcal{CT}$, then there is no $(f, t') \in \mathcal{K}$,
2. for any $t \in \mathcal{T}$, the number of CorTokGen oracle queries, on input $(t, \cdot)$, is bounded by a constant,
3. the number of HonUpdate oracle queries, on input $(\cdot, \cdot, 0, \cdot)$, is bounded by a constant.

The first restriction is due to limitations in our current proof techniques: given $t' \in \mathcal{T}$, $t' \neq t^*$, the reduction can either simulate $\Delta_{t^*\to t'}$, or $sk_{f,t'}$, for any arbitrary f. Since CorTokGen requires generating $\Delta_{t^*\to t'}$, the reduction wouldn't be able to simulate $sk_{f,t'}$ as well. The last two restrictions are instead due to the security loss that the guessing strategy would otherwise lead to as the target tags of tokens, where the source tag is the challenge one, and challenge update queries made have to be guessed in advance. Since the proof is in the random oracle model, these guesses are not over the entire tag-space $\mathcal{T}$, which can be unbounded, but over the indices of the RO queries, which are bounded by a polynomial in the security parameter as the adversary needs to be efficient. As long as the number of CorTokGen-oracle queries per given source tag, and HonUpdate-oracle queries on input the challenge ciphertext, are constant, the security loss will be polynomially bounded. We will make this assumption in Theorem 4. This result can also be rephrased in the following terms: if one maintains a "recording graph" that has a node for each tag queried to the RO, and whose edges are derived from the tokens and challenge updates issued to the adversary, then the loss is given by n^δ, where n is the number of nodes in the graph, and δ is the outer degree of the graph. This result is similar to the one obtained by Fuchsbauer et al. [FKKP19] to generically obtain proxy re-encryption schemes secure against adversaries that can adaptively corrupt users from proxy re-encryption schemes secure against adversaries that cannot make adaptive user corruptions.

Theorem 4 (Security). *Let λ be the security parameter. Fix parameters q, n, m, α, σ, ρ, ρ_1, ρ_2, μ and τ as above. Then, under the above restrictions on the adversary, the* CUFE *scheme described in Fig 5 is adaptive* IND-CUFE-CPA *secure if the ALS-IPFE scheme [ALS16] is* AD-IND *secure.*

Proof. We proceed in a series of hybrids, consider $\mathcal{A}$ to be a PPT adversary, and λ to be the security parameter. We denote by $\mathbf{Adv}_{\mathrm{Game}_i}(\mathcal{A})$ the advantage of $\mathcal{A}$ in Game i. Let Q_{h} be the number of random-oracle queries made by the adversary, Q_{t} be the maximum number of TokGen-oracle queries of the form (t, t_i) for any fixed tag t, and Q_{u} be the maximum number of Update-oracle queries on input the challenge ciphertext. We will assume, without loss of generality, that any adversary making key generation queries of the form $(\vec{y}, t)$, update queries of the form $(t, t', \cdot, \cdot)$, or token generation queries of the form (t, t') will first query the random oracle H on t and t' (we can make this assumption because for every adversary $\mathcal{A}$, we can compile it into an adversary $\mathcal{A}'$ that exhibits this behavior).

Game_0. This is the original IND-CUFE-CPA game.

Game_1. This is the same as previous game, except that we guess the tag t^* which will be used for the challenge messages. Instead of guessing directly t^* among the set of tags $\mathcal{T}$, which would incur an exponential loss, we guess the index of the random-oracle query in which the adversary queries H to get $\mathbf{H}_{t^*,1}$ and $\mathbf{D}_{t^*}$. If the guess is incorrect, we abort. This results in a $\frac{1}{Q_{\mathrm{h}}}$ security loss.

Game_2. This is the same as previous game, except that we guess for which tags t' the adversary will query an update token of the form $(\Delta_{t^*\to t',1}, \Delta_{t^*\to t',2})$. If the guess was incorrect, we abort. As above, instead of guessing directly the tag t' among the set of tags $\mathcal{T}$, which would incur an exponential loss, we guess the indices of the random-oracle query in which the adversary queries H to get $\mathbf{H}_{t',2}$ and $\mathbf{D}_{t'}$. This will result in a $\binom{Q_{\mathrm{h}}-1}{Q_{\mathrm{t}}}^{-1}$ security loss.

Game_3. This is the same as previous game, except that we guess for which tags t' the adversary will query the Update-oracle on input the challenge ciphertext. As above, instead of guessing directly the tag t' among the set of tags $\mathcal{T}$, which would incur in an exponential loss, we guess the indices of the random-oracle query in which the adversary queries H to get $\mathbf{H}_{t',2}$ and $\mathbf{D}_{t'}$. If the guess is incorrect, we abort. This results in a $\binom{Q_\mathrm{h}-Q_\mathrm{t}-1}{Q_\mathrm{u}}^{-1}$ security loss.

From now on, let $\mathcal{H} = \{t_1, \cdots, t_{Q_\mathrm{h}}\}$ be the list of random-oracle queries made by the adversary. Let $i^* \in [Q_\mathrm{h}]$ be the index of the query corresponding to the challenge tag, i.e., $t_{i^*} = t^*$. Let $\mathcal{QT}$ be the list of indices $\{i_k\}_{k \le Q_\mathrm{t}}$ for which the adversary will query an update token from the challenge tag t^*, and let $\mathcal{QU}$ be the list of indices $\{j_k\}_{k \le Q_\mathrm{u}}$ for which the adversary will query the Update-oracle for a ciphertext encrypted under the challenge tag t^*.

Game_4. This is the same as previous game, except for the following modifications. For each of $i_k \in \mathcal{QT}$, we sample $\mathbf{X}_{t^*,t_{i_k}}, \mathbf{Y}_{t^*,t_{i_k}} \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m},\rho}$, and $\Delta_{t^*\to t,2} \leftarrow \mathcal{D}_{\mathbb{Z}^{2m\times m},\rho}$. Then, we set $H_2(t_{i_k}) := \mathbf{B}_{t_{i_k},2} := \mathbf{A}\mathbf{X}_{t^*,t_{i_k}} + \mathbf{B}_{t^*,1}\mathbf{Y}_{t^*,t_{i_k}}$ and $H_3(t_{i_k}) := \mathbf{D}_{t_{i_k}} = \mathbf{H}_{t^*,1}\Delta_{t^*\to t,2} + \mathbf{D}_{t^*}$. When the adversary queries the $\mathsf{CorTokGen}$ oracle on input (t, t') we return

$$\Delta_{t^*\to t,1} := \left[\begin{array}{c|c}\mathbf{I}_m & \mathbf{X}_{t^*,t_{i_k}} \\ \hline \mathbf{0} & \mathbf{Y}_{t^*,t_{i_k}}\end{array}\right] \quad \text{and} \quad \Delta_{t^*\to t,2},$$

to the adversary. The rest of the game is as before. By Lemma 9, each of the token $(\Delta_{t^*\to t,1}, \Delta_{t^*\to t,2})$ is distributed statistically close to the previous game.

Game_5. This is the same as previous game, except for the following modifications. For all $i \in [Q_\mathrm{h}], i \ne i^*$, we sample $(\mathbf{B}_{t_i,1}, \mathbf{T}_{\mathbf{B}_{t_i,1}}) \leftarrow \mathsf{TrapGen}(1^n, 1^m)$ and set $H_1(t_i) := \mathbf{B}_{t_i,1}$. Whenever the adversary makes a query to the $\mathsf{CorTokGen}$ oracle of the form (t_i, t), we reply using $\mathbf{T}_{\mathbf{B}_{t_i,1}}$ instead of $\mathbf{T}_\mathbf{A}$:

– sample $\mathbf{X}_{t_i,t} \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m},\rho}$, run $\mathbf{Y}_{t',t} \leftarrow \mathsf{SamplePre}(\mathbf{B}_{t_i,1}, \mathbf{T}_{\mathbf{B}_{t_i,1}}, \mathbf{B}_{t,2} - \mathbf{A}\mathbf{X}_{t_i,t}, \rho)$, and $\mathbf{R}_{t_i\to t,2} \leftarrow \mathsf{SampleLeft}(\mathbf{B}_{t_i,1}, \mathbf{T}_{\mathbf{B}_{t_i,1}}, \mathbf{A}, \mathbf{D}_t - \mathbf{D}_{t_i}, \rho)$. Return

$$\Delta_{t_i\to t,1} := \left[\begin{array}{c|c}\mathbf{I}_m & \mathbf{X}_{t',t} \\ \hline \mathbf{0} & \mathbf{Y}_{t',t}\end{array}\right] \quad \text{and} \quad \Delta_{t_i\to t,2} := \left[\begin{array}{c|c}\mathbf{0} & \mathbf{I}_m \\ \hline \mathbf{I}_m & \mathbf{0}\end{array}\right] \cdot \mathbf{R}_{t_i\to t,2}.$$

The rest of the game is as before. Notice that, by the invariance under permutation of the Gaussian distribution, we have that $\Delta_{t_i\to t,2} \leftarrow \mathcal{D}_{\mathbb{Z}^{2m\times m},\rho}$. Moreover,

$$\mathbf{H}_{t_i,1}\Delta_{t_i\to t,2} = (\mathbf{A}|\mathbf{B}_{t_i,1}) \left[\begin{array}{c|c}\mathbf{0} & \mathbf{I}_m \\ \hline \mathbf{I}_m & \mathbf{0}\end{array}\right] \cdot \mathbf{R}_{t_i\to t,2} = (\mathbf{B}_{t_i,1}|\mathbf{A})\mathbf{R}_{t_i\to t,2} = \mathbf{D}_t - \mathbf{D}_{t_i},$$

as expected. Applying again Lemma 9, we also obtain that the distribution of $\mathsf{CorTokGen}$-oracle's replies is statistically close to that of Game_4.

Game_6. This is the same as previous game, except for the following modifications. Now, for all $i \notin \{i_1, \ldots, i_{Q_\mathrm{t}}\} \cup \{j_1, \ldots, j_{Q_\mathrm{u}}\}$, we sample $(\mathbf{B}_{t_i,2}, \mathbf{T}_{\mathbf{B}_{t_i,2}}) \leftarrow \mathsf{TrapGen}(1^n, 1^m)$ and set $H_2(t_i) := \mathbf{B}_{t_i,2}$. Whenever the adversary makes a query to the KeyGen' oracle of the form $(t_i, \vec{y})$, with $i \notin \{i_1, \ldots, i_{Q_\mathrm{t}}\} \cup \{j_1, \ldots, j_{Q_\mathrm{u}}\} \cup \{i^*\}$, we reply using $\mathbf{T}_{\mathbf{B}_{t_i,1}}$ and $\mathbf{T}_{\mathbf{B}_{t_i,2}}$ instead of $\mathbf{T}_\mathbf{A}$ (recall that $\mathbf{T}_{\mathbf{B}_{t_i,1}}$ was already introduced in the previous game for all $i \ne i^*$):

– for $\ell = 1, 2$, run $\mathsf{SampleLeft}(\mathbf{B}_{t_i,\ell}, \mathbf{T}_{\mathbf{B}_{t_i,\ell}}, \mathbf{A}, \mathbf{D}_{t_i}, \rho_\ell)$ to obtain $\mathbf{R}_{t_i,\ell}$. Return

$$\mathbf{Z}_{t_i,\ell} := \left[\begin{array}{c|c}\mathbf{0} & \mathbf{I}_m \\ \hline \mathbf{I}_m & \mathbf{0}\end{array}\right] \cdot \mathbf{R}_{t_i,\ell}.$$

The rest of the game is as before. Notice that, by the invariance under permutation of the Gaussian distribution, we have that $\mathbf{Z}_{t_i,\ell} \leftarrow \mathcal{D}_{\mathbb{Z}^{2m\times 2m},\rho_\ell}$. Moreover,

$$\mathbf{H}_{t_i,\ell_i}\mathbf{Z}_{t_i,\ell_i} = (\mathbf{A}|\mathbf{B}_{t_i,\ell_i}) \left[\begin{array}{c|c}\mathbf{0} & \mathbf{I}_m \\ \hline \mathbf{I}_m & \mathbf{0}\end{array}\right] \mathbf{R}_{t_i,\ell_i} = (\mathbf{B}_{t_i,\ell_i}|\mathbf{A})\mathbf{R}_{t_i,\ell_i} = \mathbf{D},$$

as expected. Therefore, the distribution KeyGen'-oracle's replies is, by Lemma 9, statistically close to that of Game_5.

Game_7**.** This is the same as previous game, except for the following modifications. We modify how Enc'- and $\mathsf{HonUpdate}$-oracles are handled for ciphertexts different from the challenge one. Every time the adversary makes a query to the Enc'-oracle of the form $(\vec{x}, t)$, we return $(\mathsf{ct}_{t,1,1}, \mathsf{ct}_{t,1,2}) \leftarrow \mathsf{Enc}(mpk, t, \vec{x})$, add $(\mathsf{c}, C_t, t, \vec{x})$ to $\mathcal{C}$, and increment c. Whenever the adversary makes a query to the $\mathsf{HonUpdate}$-oracle of the form $(t, t', i, \cdot)$, we check if $(\cdot, t, t', \cdot)$ is in $\mathcal{HT}$ and if $(i, \cdot, t, \vec{x})$ is in $\mathcal{C}$ for some $\vec{x} \in \mathbb{Z}_q^m$. If so, we sample $\vec{r} \leftarrow \mathbb{Z}_q^n$, $\vec{g_1} \leftarrow \mathcal{D}_{\mathbb{Z}^{2m},\tau}$, $\vec{g_2} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\tau}$, and return $(\mathsf{ct}_{t',2,1}, \mathsf{ct}_{t',2,2})$, where

$$\mathsf{ct}_{t',2,1} := \mathbf{H}_{t',2}^\top \vec{r} + \vec{g}_1, \qquad \mathsf{ct}_{t',2,2} := \mathbf{D}_{t'}^\top \vec{r} + \vec{g}_2 + \left\lfloor \frac{q}{K} \right\rfloor \cdot \vec{x},$$

otherwise we return $\perp$. By the Smudging Lemma 12, since the parameter of the Gaussian distribution from which $\vec{f_1}$ and $\vec{f_2}$ are sampled is superpolynomially bigger than the norm of $\Delta_{t\to t',1}^\top \vec{f}$ and $\vec{e}_2 + \vec{e}_3 + \Delta_{t\to t',2}^\top \vec{f}$, we get that

$$\mathsf{SD}\left(\mathcal{D}_{\mathbb{Z}^n,\tau}, \mathcal{D}_{\mathbb{Z},\tau,\Delta_{t\to t',1}^\top \vec{f}}\right), \mathsf{SD}\left(\mathcal{D}_{\mathbb{Z}^n,\tau}, \mathcal{D}_{\mathbb{Z},\tau,\vec{e}_2+\vec{e}_3+\Delta_{t\to t',2}^\top \vec{f}}\right) \le \frac{1}{\lambda^{\omega(1)}},$$

where we used again Lemma 9 to bound the norm of $\Delta_{t\to t',1}^\top \vec{f}$ and $\vec{e}_2 + \vec{e}_3 + \Delta_{t\to t',2}^\top \vec{f}$. Therefore, the distribution of Enc'- and $\mathsf{HonUpdate}$-oracle's replies is statistically close to that of Game_6.

Game_8**.** The only queries for which we still need the main secret key $\mathbf{T}_\mathbf{A}$ are the $\mathsf{HonUpdate}$-oracle queries on input the challenge ciphertext, and the functional secret key queries for the challenge tag t^* (with $\ell = 1$) and the tags t_{j_k} with $\{j_k\}_{k\le Q_\mathrm{u}}$ (for $\ell = 2$). We now perform a reduction to the security of the ALS [ALS16] encryption scheme. We reduce to the $\mathsf{AD\text{-}IND}$ security of ALS. We first obtain from the challenger public keys $\mathbf{A}_\mathrm{ALS}$, $\mathbf{D}_\mathrm{ALS}$. Now, equipped with the knowledge of t^*, we define Game_8 to be the same as Game_7, except for the following changes:

- The matrix $\mathbf{A}$ is replaced with $\mathbf{A}_\mathrm{ALS}$ instead of being generated with $\mathsf{TrapGen}$.
- We sample $\mathbf{S}_{t^*} \leftarrow \{\pm1\}^{m\times m}$ and $\mathbf{Z}_{t^*} \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m},\rho_1}$, program $H_1(t^*) := \mathbf{A}\mathbf{S}_{t^*}$ and set $H_3(t^*) := \mathbf{D}_{t^*} := \mathbf{D}_\mathrm{ALS} + \mathbf{A}\mathbf{S}_{t^*}\mathbf{Z}_{t^*}$.
- Similarly, for each $k \in [Q_\mathrm{u}]$, we sample $\mathbf{S}_{t_{j_k}} \leftarrow \{\pm1\}^{m\times m}$ and $\mathbf{R}_{t_{j_k}}, \mathbf{Z}_{t_{j_k}} \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m},\rho_2}$, program $H_2(t_{j_k}) := \mathbf{A}\mathbf{S}_{t_{j_k}}$ and set $H_3(t_{j_k}) := \mathbf{D}_{t_{j_k}} = \mathbf{D}_\mathrm{ALS} + \mathbf{A}\mathbf{R}_{t_{j_k}} + \mathbf{A}\mathbf{S}_{t_{j_k}}\mathbf{Z}_{t_{j_k}}$
- For key queries of the form $(t, \vec{y})$, we forward $\vec{y}$ to the challenger of the $\mathsf{AD\text{-}IND}$ security of ALS, which replies with $sk_{\vec{y}} = \mathbf{Z}_\mathrm{ALS} \cdot \vec{y}$, where $\mathbf{Z}_\mathrm{ALS}$ is the main secret key of the ALS scheme. If $t = t^*$, we set

$$sk_{t^*,1,\vec{y}} := \begin{pmatrix} sk_{\vec{y}} \\ \mathbf{Z}_{t^*}\vec{y} \end{pmatrix},$$

  and using $\mathbf{T}_{\mathbf{B}_{t^*,2}}$ we compute $sk_{t^*,2,\vec{y}}$. If $t = t_{j_k}$ for some $k \in [Q_\mathrm{u}]$, then we set

$$sk_{t_{j_k},2,\vec{y}} := \begin{pmatrix} sk_{\vec{y}} + \mathbf{R}_{t_{j_k}}\vec{y} \\ \mathbf{Z}_{t^*}\vec{y} \end{pmatrix},$$

  and using $\mathbf{T}_{\mathbf{B}_{t_{j_k},1}}$ we compute $sk_{t_{j_k},1,\vec{y}}$. One forwards both to the adversary.
- When the adversary finally submits a challenge $(\vec{x}_0, \vec{x}_1)$, we forward it to the ALS challenger, which replies with $\mathsf{ct} = (\mathsf{ct}_1^\mathrm{ALS}, \mathsf{ct}_2^\mathrm{ALS})$. We compute

$$\mathsf{ct}_{t^*,1} = (\mathsf{ct}_1^\mathrm{ALS} | (\mathbf{S}_{t^*})^\top \cdot \mathsf{ct}_1^\mathrm{ALS}),$$
$$\mathsf{ct}_{t^*,2} = \mathsf{ct}_2^\mathrm{ALS} + (\mathbf{R}_{t^*} + \mathbf{S}_{t^*}\mathbf{Z}_{t^*})^\top \cdot \mathsf{ct}_1^\mathrm{ALS} + \mathsf{NoiseGen}((\mathbf{R}_{t^*} + \mathbf{S}_{t^*}\mathbf{Z}_{t^*})^\top, s),$$

  forward $(\mathsf{ct}_{t^*,1}, \mathsf{ct}_{t^*,2})$ back to the adversary. (The properties of the algorithm $\mathsf{NoiseGen}$ are recalled in Lemma 13 from Appendix A.4.)[10]
- Whenever the adversary queries the $\mathsf{HonUpdate}$ oracle on input the challenge ciphertext $(\mathsf{ct}_{t^*,1}, \mathsf{ct}_{t^*,2})$ and target tag t_{j_k}, we compute

$$\mathsf{ct}_{t_{j_k},1} = (\mathsf{ct}_1^\mathrm{ALS} | (\mathbf{S}_{t_{j_k}})^\top \cdot \mathsf{ct}_1^\mathrm{ALS}) + \mathbf{H}_{t_{j_k},2}^\top \vec{r} + \vec{g}_1,$$
$$\mathsf{ct}_{t_{j_k},2} = \mathsf{ct}_2^\mathrm{ALS} + (\mathbf{R}_{t_{j_k}} + \mathbf{S}_{t_{j_k}}\mathbf{Z}_{t_{j_k}})^\top \cdot \mathsf{ct}_1^\mathrm{ALS} + \mathbf{D}_{t_{j_k}}^\top \vec{r} + \vec{g}_2,$$

  and forward it to the adversary.

[10] Notice that it is possible to rely on the Smudging Lemma here as well. To simplify the proof we use the properties of $\mathsf{NoiseGen}$, as done by [ACGU20], and directly refer to their security proof.

In this game, the advantage of the adversary is upper bounded by the advantage of breaking the ALS scheme, i.e., that $\mathbf{Adv}_{\text{Game}_8}(\mathcal{A}) \leq \mathbf{Adv}_{\text{ALS}}(\mathcal{A})$. It remains to show that Game_8 is indistinguishable from Game_7. We show that the update of the challenge ciphertext and function keys for tag t_{j_k}, with $k \in [Q_\text{u}]$, are statistically close to those obtained in Game_7. An identical argument to that used in [ACGU20] proves the same for the challenge tag t^*. We start by considering the function keys. Since the parameter of the Gaussian distribution from which $\mathbf{R}_{t_{j_k}}$ is sampled is superpolynomially bigger than the norm of $\mathbf{Z}_{\text{ALS}}$, by the Smudging Lemma 12 we have that $sk_{\vec{y}} + \mathbf{R}_{t_{j_k}}$ is distributed statistically close to $\mathcal{D}_{\mathbb{Z}^{m\times m},\rho_2}$. Moreover, we have that

$$\begin{aligned}\mathbf{H}_{t_{j_k},2} \cdot sk_{t_{j_k},2,\vec{y}} &= (\mathbf{A}|\mathbf{A}\mathbf{S}_{t_{j_k}}) \begin{pmatrix} sk_{\vec{y}} + \mathbf{R}_{t_{j_k}}\vec{y} \\ \mathbf{Z}_{t_{j_k}}\vec{y} \end{pmatrix} \\ &= \mathbf{A}sk_{\vec{y}} + \mathbf{A}\mathbf{R}_{t_{j_k}}\vec{y} + \mathbf{A}\mathbf{S}_{t_{j_k}}\mathbf{Z}_{t_{j_k}}\vec{y} = \mathbf{D}_{t_{j_k}}\vec{y},\end{aligned}$$

as expected. As far as the update of the challenge ciphertext is concerned, as before, since the parameter of the distribution from which $\vec{g}_2$ is drawn is superpolynomially bigger than the norm of the other error terms in the expression of $\mathsf{ct}_{t_{j_k},2}$, again by the Smudging Lemma 12, we obtain that the distribution of the ciphertext so obtained is statistically close to that of Game_7.

Putting everything together, we obtain that

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ind\text{-}cufe\text{-}cpa}}_{\mathsf{CUFE},\mathcal{A}}(\lambda, \mathcal{Y}) &\leq Q_\text{h}\binom{Q_\text{h}-1}{Q_\text{t}}\binom{Q_\text{h}-Q_\text{t}-1}{Q_\text{u}} \cdot \mathbf{Adv}_{\text{ALS}}(\mathcal{A}) + \mathsf{negl}(n) \\ &\leq Q_\text{h}^{(Q_\text{t}+Q_\text{u}+1)} \cdot \mathbf{Adv}_{\text{ALS}}(\mathcal{A}) + \mathsf{negl}(\lambda).\end{aligned}$$

□

6 Conclusion

In this work we proposed ciphertext updatable functional encryption (CUFE), a variant of functional encryption which allows switching ciphertexts produced with respect to one tag to one under another tag using an update token for this tag pair. We have provided practical motivation for such a primitive and then defined an (adaptive) security notion in the indistinguishability setting for CUFE. We presented two constructions, where the first construction is a generic construction of CUFE for any functionality, which can also be extended to predicates other than the equality testing on tags. This construction is based on (probabilistic) indistinguishability obfuscation (iO) and is proven to achieve (fully) selective security. The second construction is a (plausibly) post-quantum CUFE for the inner product functionality that relies on standard assumptions from lattices. The lattice-based construction achieves the stronger adaptive security notion, albeit with certain restrictions on the validity of the adversary and bound on the number of oracle queries. We leave it as an interesting open problem to construct a CUFE scheme that satisfies our adaptive security model without any further restrictions or bound on the number of oracle queries. Moreover, we consider it an interesting open problem to construct practical CUFE schemes for a richer class of functionalities, e.g., quadratic functions, which can further broaden the scope of application.

Acknowledgements. This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n°871473 (KRAKEN), by the Austrian Science Fund (FWF) and netidee SCIENCE via grant P31621-N38 (PROFET) and FWF via grant W1255-N23.

References

ABB10. Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 553–572. Springer, Heidelberg, May / June 2010.

ABDP15. Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple functional encryption schemes for inner products. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 733–751. Springer, Heidelberg, March / April 2015.

ABDP16. Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Better security for functional encryption for inner product evaluations. Cryptology ePrint Archive, Report 2016/011, 2016. https://eprint.iacr.org/2016/011.

ABG19. Michel Abdalla, Fabrice Benhamouda, and Romain Gay. From single-input to multi-client inner-product functional encryption. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 552–582. Springer, Heidelberg, December 2019.

ABKW19. Michel Abdalla, Fabrice Benhamouda, Markulf Kohlweiss, and Hendrik Waldner. Decentralizing inner-product functional encryption. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part II*, volume 11443 of *LNCS*, pages 128–157. Springer, Heidelberg, April 2019.

ABM+20. Michel Abdalla, Florian Bourse, Hugo Marival, David Pointcheval, Azam Soleimanian, and Hendrik Waldner. Multi-client inner-product functional encryption in the random-oracle model. In Clemente Galdi and Vladimir Kolesnikov, editors, *SCN 20*, volume 12238 of *LNCS*, pages 525–545. Springer, Heidelberg, September 2020.

ABSV15. Prabhanjan Ananth, Zvika Brakerski, Gil Segev, and Vinod Vaikuntanathan. From selective to adaptive security in functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 657–677. Springer, Heidelberg, August 2015.

ACF+18. Michel Abdalla, Dario Catalano, Dario Fiore, Romain Gay, and Bogdan Ursu. Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 597–627. Springer, Heidelberg, August 2018.

ACGU20. Michel Abdalla, Dario Catalano, Romain Gay, and Bogdan Ursu. Inner-product functional encryption with fine-grained access control. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 467–497. Springer, Heidelberg, December 2020.

ACH20. Thomas Agrikola, Geoffroy Couteau, and Dennis Hofheinz. The usefulness of sparsifiable inputs: How to avoid subexponential iO. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part I*, volume 12110 of *LNCS*, pages 187–219. Springer, Heidelberg, May 2020.

AFGH05. Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. In *NDSS 2005*. The Internet Society, February 2005.

AFS21. Miguel Ambrona, Dario Fiore, and Claudio Soriente. Controlled functional encryption revisited: Multi-authority extensions and efficient schemes for quadratic functions. *PoPETs*, 2021(1):21–42, January 2021.

AGRW17. Michel Abdalla, Romain Gay, Mariana Raykova, and Hoeteck Wee. Multi-input inner-product functional encryption from pairings. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 601–626. Springer, Heidelberg, April / May 2017.

AJS18. Prabhanjan Ananth, Aayush Jain, and Amit Sahai. Indistinguishability obfuscation without multilinear maps: iO from LWE, bilinear maps, and weak pseudorandomness. Cryptology ePrint Archive, Report 2018/615, 2018. https://eprint.iacr.org/2018/615.

ALS16. Shweta Agrawal, Benoît Libert, and Damien Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 333–362. Springer, Heidelberg, August 2016.

BBL17. Fabrice Benhamouda, Florian Bourse, and Helger Lipmaa. CCA-secure inner-product functional encryption from projective hash functions. In Serge Fehr, editor, *PKC 2017, Part II*, volume 10175 of *LNCS*, pages 36–66. Springer, Heidelberg, March 2017.

BBS98. Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 127–144. Springer, Heidelberg, May / June 1998.

BFM88. Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, page 103–112, New York, NY, USA, 1988. Association for Computing Machinery.

BGI+12. Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6:1–6:48, 2012.

BSW11. Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 253–273. Springer, Heidelberg, March 2011.

BSW16. Mihir Bellare, Igors Stepanovs, and Brent Waters. New negative results on differing-inputs obfuscation. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 792–821. Springer, Heidelberg, May 2016.

CCL+14. Nishanth Chandran, Melissa Chase, Feng-Hao Liu, Ryo Nishimaki, and Keita Xagawa. Re-encryption, functional re-encryption, and multi-hop re-encryption: A framework for achieving obfuscation-based security and instantiations from lattices. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 95–112. Springer, Heidelberg, March 2014.

CDG+18. Jérémy Chotard, Edouard Dufour Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. Decentralized multi-client functional encryption for inner product. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 703–732. Springer, Heidelberg, December 2018.

CLT18. Guilhem Castagnos, Fabien Laguillaumie, and Ida Tucker. Practical fully secure unrestricted inner product functional encryption modulo p. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 733–764. Springer, Heidelberg, December 2018.

CLTV15. Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. Obfuscation of probabilistic circuits and applications. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 468–497. Springer, Heidelberg, March 2015.

Coh19. Aloni Cohen. What about bob? The inadequacy of CPA security for proxy reencryption. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part II*, volume 11443 of *LNCS*, pages 287–316. Springer, Heidelberg, April 2019.

CZY19. Yuechen Chen, Linru Zhang, and Siu-Ming Yiu. Practical attribute based inner product functional encryption from simple assumptions. Cryptology ePrint Archive, Report 2019/846, 2019. https://eprint.iacr.org/2019/846.

DHRW16. Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. Spooky encryption and its applications. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 93–122. Springer, Heidelberg, August 2016.

DKL+18. David Derler, Stephan Krenn, Thomas Lorünser, Sebastian Ramacher, Daniel Slamanig, and Christoph Striecks. Revisiting proxy re-encryption: Forward secrecy, improved security, and applications. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 219–250. Springer, Heidelberg, March 2018.

DM14. Léo Ducas and Daniele Micciancio. Improved short lattice signatures in the standard model. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 335–352. Springer, Heidelberg, August 2014.

DN21. Nico Döttling and Ryo Nishimaki. Universal proxy re-encryption. In Juan Garay, editor, *PKC 2021, Part I*, volume 12710 of *LNCS*, pages 512–542. Springer, Heidelberg, May 2021.

DP19. Edouard Dufour Sans and David Pointcheval. Unbounded inner-product functional encryption with succinct keys. In Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung, editors, *ACNS 19*, volume 11464 of *LNCS*, pages 426–441. Springer, Heidelberg, June 2019.

FKKP19. Georg Fuchsbauer, Chethan Kamath, Karen Klein, and Krzysztof Pietrzak. Adaptively secure proxy re-encryption. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part II*, volume 11443 of *LNCS*, pages 317–346. Springer, Heidelberg, April 2019.

FL19. Xiong Fan and Feng-Hao Liu. Proxy re-encryption and re-signatures from lattices. In Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung, editors, *ACNS 19*, volume 11464 of *LNCS*, pages 363–382. Springer, Heidelberg, June 2019.

FMNV14. Sebastian Faust, Pratyay Mukherjee, Jesper Buus Nielsen, and Daniele Venturi. Continuous non-malleable codes. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 465–488. Springer, Heidelberg, February 2014.

FS16. Somchart Fugkeaw and Hiroyuki Sato. Updating policies in cp-abe-based access control: An optimized and secure service. In Marco Aiello, Einar Broch Johnsen, Schahram Dustdar, and Ilche Georgievski, editors, *ESOCC 2016*, volume 9846 of *Lecture Notes in Computer Science*, pages 3–17. Springer, 2016.

GGH+13. Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013.

GGHW14. Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 518–535. Springer, Heidelberg, August 2014.

GOS06. Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 339–358. Springer, Heidelberg, May / June 2006.

GPV08. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008.

GVW13. Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 545–554. ACM Press, June 2013.

JLMS19. Aayush Jain, Huijia Lin, Christian Matt, and Amit Sahai. How to leverage hardness of constant-degree expanding polynomials overa $\mathbb{R}$ to build $i\mathcal{O}$. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 251–281. Springer, Heidelberg, May 2019.

JLS19. Aayush Jain, Huijia Lin, and Amit Sahai. Simplifying constructions and assumptions for $i\mathcal{O}$. Cryptology ePrint Archive, Report 2019/1252, 2019. https://eprint.iacr.org/2019/1252.

Kaw15. Yutaka Kawai. Outsourcing the re-encryption key generation: Flexible ciphertext-policy attribute-based proxy re-encryption. In Javier López and Yongdong Wu, editors, *ISPEC 2015*, volume 9065 of *Lecture Notes in Computer Science*, pages 301–315. Springer, 2015.

KPC+20. Vlasis Koutsos, Dimitrios Papadopoulos, Dimitris Chatzopoulos, Sasu Tarkoma, and Pan Hui. Agora: A privacy-aware data marketplace. In *ICDCS*, pages 1211–1212. IEEE, 2020.

KV11. Jonathan Katz and Vinod Vaikuntanathan. Round-optimal password-based authenticated key exchange. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 293–310. Springer, Heidelberg, March 2011.

KY16. Shuichi Katsumata and Shota Yamada. Partitioning via non-linear polynomial functions: More compact IBEs from ideal lattices and bilinear maps. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 682–712. Springer, Heidelberg, December 2016.

LLW21. Qiqi Lai, Feng-Hao Liu, and Zhedong Wang. New lattice two-stage sampling technique and its applications to functional encryption - stronger security and smaller ciphertexts. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 498–527. Springer, Heidelberg, October 2021.

LT19. Benoît Libert and Radu Titiu. Multi-client functional encryption for linear functions in the standard model from LWE. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 520–551. Springer, Heidelberg, December 2019.

MR04. Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th FOCS*, pages 372–381. IEEE Computer Society Press, October 2004.

MSH+19. Tilen Marc, Miha Stopar, Jan Hartman, Manca Bizjak, and Jolanda Modic. Privacy-enhanced machine learning with functional encryption. In Kazue Sako, Steve Schneider, and Peter Y. A. Ryan, editors, *ESORICS 2019, Part I*, volume 11735 of *LNCS*, pages 3–21. Springer, Heidelberg, September 2019.

NAP+14. Muhammad Naveed, Shashank Agrawal, Manoj Prabhakaran, XiaoFeng Wang, Erman Ayday, Jean-Pierre Hubaux, and Carl A. Gunter. Controlled functional encryption. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014*, pages 1280–1291. ACM Press, November 2014.

NPP22. Ky Nguyen, Duong Hieu Phan, and David Pointcheval. Multi-client functional encryption with fine-grained access control. Cryptology ePrint Archive, Report 2022/215, 2022. https://eprint.iacr.org/2022/215.

NY90. Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*, pages 427–437. ACM Press, May 1990.

O'N10. Adam O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. https://eprint.iacr.org/2010/556.

PD21. Tapas Pal and Ratna Dutta. Attribute-based access control for inner product functional encryption from LWE. In Patrick Longa and Carla Ràfols, editors, *LATINCRYPT 2021*, volume 12912 of *LNCS*, pages 127–148. Springer, Heidelberg, October 2021.

Reg05. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

SW05. Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer, Heidelberg, May 2005.

Wee17. Hoeteck Wee. Attribute-hiding predicate encryption in bilinear groups, revisited. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 206–233. Springer, Heidelberg, November 2017.

Appendix

A Additional Preliminaries

A.1 Public-Key Encryption

Definition 6. *A public-key encryption scheme* $\Sigma = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *with message space* $\mathcal{M}$ *consists of the following PPT algorithms:*
- $\Sigma.\mathsf{KeyGen}(1^\lambda)$*, on input a security parameter* 1^λ*, outputs a secret/public key pair* (sk, pk)*.*
- $\Sigma.\mathsf{Enc}(pk, m)$*, on input a public key* pk *and a message* m*, outputs a ciphertext* c*.*
- $\Sigma.\mathsf{Dec}(sk, c)$*, on input a secret key* sk *and a ciphertext* c*, outputs a message* $m \in \mathcal{M} \cup \{\perp\}$*.*

We say that an encryption scheme Σ is perfectly correct if for all $\lambda \in \mathbb{N}$, for all $(sk, pk) \leftarrow \Sigma.\mathsf{KeyGen}(1^\lambda)$ and for all $m \in \mathcal{M}$ it holds that $\Sigma.\mathsf{Dec}(sk, \Sigma.\mathsf{Enc}(pk, m)) = m$.

Next, we recall the standard notion of indistinguishability under chosen plaintext attacks ($\mathsf{IND\text{-}CPA}$ security).

Definition 7 (IND-CPA). *A public-key encryption scheme* Σ *is* $\mathsf{IND\text{-}CPA}$ *secure, if for all PPT adversaries* A *it holds that*

$$\mathsf{Adv}^{\mathsf{ind-cpa}}_{\Sigma,A}(\lambda) := \left| \Pr\left[\begin{array}{l} (sk, pk) \leftarrow \Sigma, b \leftarrow \{0,1\}, \\ (m_0, m_1, \mathsf{st}) \leftarrow A(pk), b^* \leftarrow A(\Sigma.\mathsf{Enc}(pk, m_b), \mathsf{st}) : \\ b = b^* \end{array}\right] - \frac{1}{2} \right|,$$

is negligible.

A.2 Non-Interactive Zero-Knowledge

Let R be an efficiently computable binary relation, where for pairs $(x, w) \in R$ we call x the statement and w the witness. Let L be the language consisting of statements in R. A non-interactive zero-knowledge (NIZK) proof system [BFM88, GOS06] for a language L allows proving that some statements are in L without leaking information about the corresponding witnesses in a non-interactive manner. We note that we require the proof system to support labels. This can be done by extending the algorithms of the proof system to also take a public label ℓ as input as described in [KV11, FMNV14].

Definition 8 (Labeled Statistically Simulation Sound NIZK Proof System). *A labeled statistically simulation sound non-interactive zero-knowledge (ℓ-SSS-NIZK) proof system* Π *for a language* $L \in \mathsf{NP}$ *(with witness relation* R*) with label set* $\mathcal{L}$ *is a tuple of PPT algorithms* $\Pi = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$*, such that:*
- $\Pi.\mathsf{Setup}(1^\lambda)$*, on input a security parameter* 1^λ*, outputs a common reference string* crs*.*
- $\Pi.\mathsf{Prove}(\mathsf{crs}, \ell, x, w)$*, on input* crs*, a label* ℓ*, a statement* x *and a witness* w*, outputs a proof* π*.*
- $\Pi.\mathsf{Verify}(\mathsf{crs}, \ell, x, \pi)$*, on input* crs*, a label* ℓ*, a statement* x *and a proof* π*, outputs either* 1 *or* 0*.*

We require Π *to meet the following properties:*

Perfect completeness. *For every* $(x, w) \in R$ *and* $\ell \in \mathcal{L}$*, we have that*

$$\Pr\left[\mathsf{crs} \leftarrow \Pi.\mathsf{Setup}(1^\lambda), \pi \leftarrow \Pi.\mathsf{Prove}(\mathsf{crs}, \ell, x, w) \colon \Pi.\mathsf{Verify}(\mathsf{crs}, \ell, x, \pi) = 1\right] = 1.$$

Statistical soundness. *For every* $x \notin L$*, and every (possibly unbounded) adversary* A*, we have that*

$$\Pr\left[\mathsf{crs} \leftarrow \Pi.\mathsf{Setup}(1^\lambda), (\ell, \pi) \leftarrow A(\mathsf{crs}, x) \colon \ell \in \mathcal{L} \ \wedge\ \Pi.\mathsf{Verify}(\mathsf{crs}, \ell, x, \pi) = 1\right] \leq \mathsf{negl}(\lambda).$$

Computational zero-knowledge. *There exists a PPT algorithm* $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ *such that for every PPT adversary* A*,*

$$\begin{aligned} \mathsf{Adv}^{\mathsf{ZK}}_{A}(\lambda) := \Big| & \Pr\left[\mathsf{crs} \leftarrow \Pi.\mathsf{Setup}(1^\lambda) \colon A^{\Pi.\mathsf{Prove}(\mathsf{crs},\cdot,\cdot,\cdot)}(\mathsf{crs}) = 1\right] \\ & - \Pr\left[(\mathsf{crs}, \tau) \leftarrow \mathsf{Sim}_1(1^\lambda) \colon A^{\mathcal{O}(\mathsf{crs},\tau,\cdot,\cdot,\cdot)}(\mathsf{crs}) = 1\right] \Big| \end{aligned}$$

is negligible in λ*, where* $\mathcal{O}(\mathsf{crs}, \tau, \cdot, \cdot, \cdot)$ *is an oracle that outputs* $\perp$ *on input* (ℓ, x, w) *when* $(x, w) \notin R$ *and outputs* $\pi \leftarrow \mathsf{Sim}_2(\mathsf{crs}, \tau, \ell, x)$ *when* $(x, w) \in R$*.*

Statistical simulation soundness. *For every statement* x *and (possibly unbounded) adversary* A*, we have that*

$$\Pr\left[\begin{array}{l}(\mathsf{crs}, \tau) \leftarrow \mathsf{Sim}_1(1^\lambda, x), (\ell', x', \pi') \leftarrow A^{\mathsf{Sim}_2(\mathsf{crs}, \tau, \cdot, \cdot)}(\mathsf{crs}): \\ (\ell', x', \pi') \notin Q \ \wedge \ x' \notin L \ \wedge \ \ell' \in \mathcal{L} \ \wedge \ \Pi.\mathsf{Verify}(\mathsf{crs}, \ell', x', \pi') = 1\end{array}\right] \leq \mathsf{negl}(\lambda),$$

where Q *is the set of all simulated queries and responses* (ℓ_i, x_i, π_i) *made by* A *to* $\mathsf{Sim}_2(\mathsf{crs}, \tau, \cdot, \cdot)$*.*

Garg et al. [GGH+13] showed how to turn any statistically sound NIZK proof system into an SSS-NIZK proof system using any non-interactive commitment scheme. However, statistical simulation-soundness can only be achieved if the number of false statements for which a valid proof exists is a priori bounded. Analogous to [GGH+13], we also only give one fake proof in the challenge ciphertext, hence, we satisfy this constraint.

A.3 Functional Encryption with Adaptive Security

We recall the adaptive variant for the security of functional encryption here.

Definition 9. *For every functional encryption scheme* FE *for functionality* $\mathcal{F}\colon \mathcal{X} \to \mathcal{Y}$*, every security parameter* λ*, every PPT adversary* A*, we define the following experiment: where* $\mathcal{O}_{\mathsf{KeyGen}}(\cdot)$

Experiment $\mathsf{Exp}^{\mathsf{ind-cpa}}_{\mathsf{FE},A}(\lambda, \mathcal{F})$
$(mpk, msk) \leftarrow \mathsf{Setup}(\lambda, \mathcal{F})$
$(x_0^*, x_1^*, \mathsf{st}) \leftarrow A^{\mathcal{O}_{\mathsf{KeyGen}}(\cdot)}(mpk, \mathsf{st})$
$b \leftarrow \{0, 1\}$
$C^* \leftarrow \mathsf{Enc}(mpk, x_b)$
$b' \leftarrow A^{\mathcal{O}_{\mathsf{KeyGen}}(\cdot)}(C^*, \mathsf{st})$
if $b = b'$ then return 1 else return 0

is an oracle that on input $f \in \mathcal{F}$*, outputs* $\mathsf{KeyGen}(msk, f)$*. Additionally, if* A *ever calls the oracle* $\mathcal{O}_{\mathsf{KeyGen}}(\cdot)$ *on an input* $f \in \mathcal{F}$*, the challenge queries* x_0^**,* x_1^* *must satisfy* $f(x_0^*) = f(x_1^*)$*. A functional encryption scheme* FE *is* $\mathsf{AD\text{-}IND}$ *secure if for every PPT adversary* A *the advantage function*

$$\mathsf{Adv}^{\mathsf{ind-cpa}}_{\mathsf{FE},A}(\lambda, \mathcal{F}) := \left|\Pr\left[\mathsf{Exp}^{\mathsf{ind-cpa}}_{\mathsf{FE},A}(\lambda, \mathcal{F}) = 1\right] - 1/2\right|,$$

is negligible in λ*.*

A.4 Lattice Preliminaries

Definition 10 ([Reg05] Learning with errors). *Let* q *be a prime,* χ *be a public distribution over* $\mathbb{Z}_q$ *and* $\vec{s}$ *be uniformly random over* $\mathbb{Z}_q^n$*. Moreover,* $\vec{s}$ *is constant across calls to oracles* $\mathcal{O}_{\vec{s}}$*, or* $\mathcal{O}_\$$*, defined below:*

- *Oracle* $\mathcal{O}_{\vec{s}}$ *outputs samples* $(\vec{a}, \langle \vec{a}, \vec{s}\rangle + e)$ *where* $\vec{a} \leftarrow \mathbb{Z}_q^n$ *and* $e \leftarrow \chi$ *are fresh and independently sampled,*
- *Oracle* $\mathcal{O}_\$$ *outputs uniformly random elements of* $\mathbb{Z}_q^n \times \mathbb{Z}_q$*.*

Define another oracle $\mathcal{O}$*, which across all calls, is either* $\mathcal{O}_{\vec{s}}$ *or* $\mathcal{O}_\$$*. The learning with errors* $LWE_{q,\chi,n}$ *problem is to distinguish with non-negligible probability, given access to oracle* $\mathcal{O}$*, whether it corresponds to* $\mathcal{O}_{\vec{s}}$ *or* $\mathcal{O}_\$$*.*

Lemma 11 ([MR04],[GPV08] Gaussian Tail Bound). *For any* n*-dimensional lattice* Λ*,* $\vec{c} \in span(\Lambda)$*, real* $\epsilon \in (0, 1)$*, and* $s \geq \eta_\epsilon(\Lambda)$*:*

$$\Pr_{\vec{x} \leftarrow \mathcal{D}_{\Lambda,s,\vec{c}}}[\|\vec{x} - \vec{c}\| > s\sqrt{n}] \leq \frac{1+\epsilon}{1-\epsilon}\frac{1}{2^n}.$$

Moreover, for any $\omega(\sqrt{\log n})$ function, there is a negligible $\epsilon(n)$ such that: $\eta_\epsilon(\mathbb{Z}) \leq \omega(\sqrt{\log n})$. In particular, when sampling integers, we have that for any $\epsilon \in (0, \frac{1}{2})$, any $s \geq \eta_\epsilon(\mathbb{Z})$, and any $t \geq \omega(\sqrt{\log n})$:

$$\Pr_{x \leftarrow \mathcal{D}_{\mathbb{Z},s,c}} [|x - c| > s \cdot t] \leq \mathsf{negl}(n).$$

Lemma 12 (Smudging Lemma). *Let $n \in \mathbb{N}$. For any real $\sigma > \omega(\sqrt{\log n})$, and any $\vec{c} \in \mathbb{Z}^n$, it holds $\mathsf{SD}(\mathcal{D}_{\mathbb{Z}^n,\sigma}, \mathcal{D}_{\mathbb{Z}^n,\sigma,\vec{c}}) \leq \|\vec{c}\|/\sigma$.*

Noise Re-randomization. The following procedure of $\mathsf{NoiseGen}(\mathbf{R}, s)$ for noise re-randomization, was described in [KY16]. $\mathsf{NoiseGen}(\mathbf{R}, s)$: given a matrix $\mathbf{R} \in \mathbb{Z}^{m \times t}$, and $s \in \mathbb{R}^+$ such that $s^2 > s_1(\mathbf{R}\mathbf{R}^\top)$, it first samples $\vec{e}_1 := \mathbf{R}\vec{e} + (s^2\mathbf{I}_m - \mathbf{R}\mathbf{R}^\top)^{\frac{1}{2}}\vec{e}'$, where $\mathbf{I}_m \in \mathbb{Z}^{m \times m}$ denotes the identity matrix, and $\vec{e} \leftarrow \mathcal{D}_\sigma^t$, and $\vec{e}' \leftarrow \mathcal{D}_{\sqrt{2}\sigma}^m$ are independent spherical continuous Gaussian noises. Then, it samples $\vec{e}_2 \leftarrow \mathcal{D}_{\mathbb{Z}^m - \vec{e}_1, s\sqrt{2}\sigma}$, and return $\vec{e}_1 + \vec{e}_2 \in \mathbb{Z}_q^m$. We have the following lemma.

Lemma 13 ([KY16] Noise Distribution). *Let $\mathbf{R} \leftarrow \mathbb{Z}^{m \times t}$ and $s \geq s_1(\mathbf{R})$. The following distributions are statistically close:*
Distribution 1: *$\vec{e} \leftarrow \mathcal{D}_{\mathbb{Z}^t,\sigma}$, and $\vec{e}' \leftarrow \mathsf{NoiseGen}(\mathbf{R}, s)$. Output $\mathbf{R}\vec{e} + \vec{e}'$.*
Distribution 2: *Output $\vec{e} \leftarrow \mathcal{D}_{\mathbb{Z}^m,2s\sigma}$.*

Lemma 14 ([ABB10] Bounding Norm of a $\{\pm 1\}^{k \times m}$ Matrix). *Let $\mathbf{R}$ be a matrix chosen uniformly at random from $\{\pm 1\}^{k \times m}$. There exists a universal constant C', for which:*

$$\Pr\left[\|\mathbf{R}\| \geq C'\sqrt{k+m}\right] < \frac{1}{e^{k+m}}.$$

Lemma 15 ([DM14] Bounding Spectral Norm of a Gaussian Matrix). *Let $\mathbf{Z} \in \mathbb{R}^{n \times m}$ be a sub-Gaussian random matrix with parameter ρ. There exists a universal constant C such that for any $t \geq 0$, we have $s_1(\mathbf{Z}) \leq C \cdot \rho(\sqrt{n} + \sqrt{m} + t)$ except with probability at most $\frac{2}{e^{\pi t^2}}$.*