(Inner-Product) Functional Encryption with Updatable Ciphertexts

Valerio Cini[1], Sebastian Ramacher[2], Daniel Slamanig[3], Christoph Striecks[2], and Erkan Tairi[4]

[1] NTT Research, Sunnyvale, CA, USA
valerio.cini@ntt-research.com
[2] AIT Austrian Institute of Technology, Vienna, Austria
{firstname.lastname}@ait.ac.at
[3] Research Institute CODE, Universität der Bundeswehr München, Munich, Germany
daniel.slamanig@unibw.de
[4] 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 allows for fine-grained access control even after a ciphertext is generated. Updating ciphertexts is carried out via so-called update tokens which a dedicated party can use to convert ciphertexts. However, allowing update tokens requires some care for the security definition. 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 tags 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 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.[5]

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

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

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, AGW20], 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 is very simple, it has already shown to be useful in privacy-preserving machine learning [MSH+19], money-laundering detection [dPP22], search in encrypted data streams [BCS22], video data analytics,[6] 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.[7] 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.[8] 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

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

[7] 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.

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

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 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). 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* and one can also think of these labels as identities (as done in [ACGU20]). 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], as in their work ciphertexts are not updatable, i.e., they do not straightforwardly provide the possibilities that a tag (identity) in a ciphertext can be changed. But as we will see, the work in [ACGU20] can serve as a starting point for our lattice-based construction. We note that 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 affirmatively via our three-fold contribution:

a) We define a new primitive dubbed ciphertext updatable functional encryption (CUFE) along 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 in the decryption key and ciphertext match, then decryption succeeds. Furthermore, we allow fresh ciphertexts to update its tag t to any other tag t' via so-called update tokens. An update token from t to t' is generated by the holder of the main secret and can only be used in the desired direction, i.e., from t to t'. In a nutshell, the distinguishing feature is that we allow changing the tag *after* a ciphertext was generated (which is not known to be achieved by existing work).

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 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, 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 honest-but-curious party[9] can take the update token to update C_t to $C_{t'}$ without learning anything about the encrypted message. 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 4.3) which ensures a simple access control for our envisioned applications.

Concerning updated ciphertexts, 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 $\mathsf{IND\text{-}CUFE\text{-}CPA}$, 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 iO. The starting point of our construction is the (semi-adaptively secure) FE construction due to Waters [Wat15], which relies on indistinguishability obfuscation (iO) and the punctured programming approach. The main ingredient of Waters' construction is a primitive called puncturable deterministic encryption (PDE), which can be constructed from puncturable PRFs using the hidden trigger mechanism of Sahai and Waters [SW14]. A PDE scheme is a symmetric and deterministic encryption scheme, which additionally has a feature that given a key k_{pde} and a pair of messages m_0, m_1, it produces a punctured key $k_{\mathsf{pde}}^{m_0,m_1}$ that can decrypt all ciphertexts except for those encrypting either m_0 or m_1[10]. Using PDE one can construct a (semi-adaptively secure) FE scheme as follows: the setup algorithm samples a puncturable PRF key k_{prf} for function F, which it sets as the main secret key, and generates an obfuscation of the program PInit, which it sets as the main public key. The program PInit takes as input a randomness r, computes a point $p = \mathsf{PRG}(r)$, derives a PDE key as $k_{\mathsf{pde}} = F(k_{\mathsf{prf}}, p)$, and outputs the pair (p, k_{pde}). The

[9] An honest-but-curious party is assumed to correctly perform the update but will not learn any information about the hidden messages.

[10] Recall that in a deterministic encryption scheme there are only two such ciphertexts.

encryption algorithm can then use the obfuscated program PInit to encrypt a message m by first sampling a randomness r, running the obfuscated program on r to receive (p, k_{pde}), and finally, computing the ciphertext as $C := (p, c := \mathsf{Enc}_{\mathsf{pde}}(k_{\mathsf{pde}}, m))$. The functional secret key sk_f, for a function f, is also created as an obfuscation of a program PKey, which has f hardcoded. This program takes as input a ciphertext $C := (p, c)$, uses p to derive the key k_{pde}, decrypts c using k_{pde} to obtain the message m, and finally, outputs $f(m)$. Hence, the decryption algorithm simply involves running the obfuscated program PKey on the ciphertext.

In order to introduce tags for the ciphertexts, a first step is to extend the PDE to its tag-based variant, that we dubbed puncturable tag-based deterministic encryption (PTDE). It works analogously to PDE, except that the ciphertexts are associated with tags and puncturing happens not only at a pair of messages m_0, m_1, but also at a tag t. Hence, a punctured key $k_{\mathsf{ptde}}^{t,m_0,m_1}$ can decrypt all ciphertexts except for those encrypting either m_0 or m_1 under the tag t. Now the challenging part is to update the ciphertexts. In order to restrict that an updated ciphertext cannot be updated anymore, we use two different puncturable PRF keys as part of the main secret key, $k_{\mathsf{prf},o}$ for the original ciphertexts and $k_{\mathsf{prf},u}$ for the updated ciphertext. Analogous to the aforedescribed construction of Waters [Wat15], these PRF keys are used to derive PTDE keys in our case. For the update operation, we now need to switch the ciphertexts encrypted under the key k_{ptde} (derived from $k_{\mathsf{prf},o}$) and tag t to a new ciphertext under the key k'_{ptde} (derived from $k_{\mathsf{prf},u}$) and tag t'. In order to do this we introduce a third program, called PUpdate, which given as input a ciphertext C_t (under a tag t) and a randomness r, first decrypts the input ciphertext C_t, and then, re-encrypts it deterministically under the new key k'_{ptde} and tag t' to produce the updated ciphertext C'_t. Due to the deterministic nature of the used cryptographic primitives, such as PTDE and puncturable PRF, we can rely solely on (plain) iO for the update operation, instead of requiring probabilistic iO [CLTV15].

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 encryption 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] of a tag-based proxy re-encryption scheme. 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 $\mathbf{A}_u = [\mathbf{A}|\mathbf{A}_{u,1}|\mathbf{A}_{u,2}]$ where $\mathbf{A}_{u,i} = -\mathbf{A}\mathbf{R}_{u,i}$ with $\mathbf{R}_{u,i}$, for $i = 1, 2$ contained in the secret key. Encryption follows a dual-Regev approach [GPV08] based on the user dependent matrix $\mathbf{A}_u$ and a random freshly sampled tag $t \in \mathcal{T}$. Re-encryption keys from 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

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

for any matrix $\mathbf{B}$. In their construction, h is a map 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 and the map h. Using as tag space $\mathcal{T}$ a large set with "unit differences" property, as introduced in [MP12], i.e, for any for any $t_i, t_j \in \mathcal{T}$, $t_i \neq t_j$, one has $h(t_i - t_j) = h(t_i) - h(t_j) \in \mathbb{Z}_q^{n \times n}$ is invertible, Fan and Liu prove their construction secure in the standard model. Their proof strategy crucially relies on the "unit differences" property together with the fact that the scheme is tag-based: the "challenge" tag, i.e. the tag associated with the challenge ciphertext, is randomly sampled at the beginning of the security game, and the public parameter are produced by embedding such "challenge" tag in them. This allows the reduction to correctly answer any allowed adversary's query, while at the same time embedding an LWE instance in the challenge ciphertext.

The setting of CUFE is however vastly different in nature as ciphertexts are not equipped with levels, there are no per-user public keys, and tags have a different meaning, in particular, they are not randomly sampled at encryption time, but are specified by the encryptor. 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. First, we move to the random oracle model in order to embed the challenge tag in the public parameters, even though in our setting such tag is specified by the encryptor, by crucially exploiting the fact that the reduction can guess the challenge tag among the random oracle queries made by the adversary. Given such modification, 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. We remark that since the reduction needs to perform guesses in order to correctly produce public parameters and answer adversary's queries, one has to make sure that the probability space over which the reduction needs to guess has at most polynomial size. In particular, such constraint will allow us to prove the lattice-based scheme secure, but only against adversary that can request at most a bounded number of update tokens per tag and honest updates of the challenge ciphertext. We discuss more in detail these restrictions in Section 5. On the other hand, while this is certainly a limitation in general, for the concrete applications we envision, one can always set parameters so that such bounds are large enough to accommodate requirements of real world scenarios.

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.

2.1 Pseudorandom Generators

We recall the definition of a (Boolean) pseudorandom generator (PRG).

Definition 1 (Pseudorandom Generator). *A stretch-$m(\cdot)$ pseudorandom generator is a (Boolean) function* $\mathsf{PRG}\colon \{0,1\}^* \to \{0,1\}^*$ *mapping n-bit inputs to $m(n)$-bit outputs (also known as the stretch) that is computable by a uniform PPT machine, and for any non-uniform PPT adversary A, there exists a negligible function* negl*, such that, for all $n \in \mathbb{N}$, the following holds*

$$\left| \Pr_{r\leftarrow\{0,1\}^n} [\, A(\mathsf{PRG}(r)) = 1 \,] - \Pr_{z\leftarrow\{0,1\}^m} [\, A(z) = 1 \,] \right| = \mathsf{negl}(\lambda).$$

2.2 Puncturable Pseudorandom Functions

Puncturable pseudorandom functions (PRFs), introduced by Sahai and Waters [SW14], are PRFs for which a key can be given out, such that it allows evaluation of the PRF on all inputs, except for a designated polynomial-size set of inputs.

Definition 2 (Puncturable PRFs [SW14]). *A puncturable family of PRFs* PRF *is given by a triple of algorithms* $(\mathsf{Gen}, F, \mathsf{Puncture})$ *and a pair of computable functions $n = n(\lambda)$ and $m = m(\lambda)$, satisfying the following conditions:*

Functionality preserved under puncturing. *For every PPT adversary A that outputs a set $S \subseteq \{0,1\}^n$, such that for all $x \in \{0,1\}^n$ where $x \notin S$, we have that:*

$$\Pr\left[F(k, x) = F(k^S, x) \mid k \leftarrow \mathsf{PRF.Gen}_F(1^\lambda), k^S \leftarrow \mathsf{PRF.Puncture}_F(k, S)\right] = 1.$$

Pseudorandom at punctured points. *For every PPT adversary (A_1, A_2), where A_1 outputs a set $S \subseteq \{0,1\}^n$ and a state σ, consider an experiment that samples $k \leftarrow \mathsf{PRF.Gen}_F(1^\lambda)$ and $k^S \leftarrow \mathsf{PRF.Puncture}_F(k, S)$, then we have*

$$\left|\Pr\left[A_2(\sigma, k^S, S, F(k,S)) = 1\right] - \Pr\left[A_2(\sigma, k^S, S, U_{m\cdot|S|}) = 1\right]\right| = \mathsf{negl}(\lambda),$$

where $F(k,S)$ denotes the concatenation of $F(k,x_1), \ldots, F(k,x_k)$, such that $S = \{x_1, \ldots, x_k\}$ is the enumeration of the elements of S in lexicographic order and U_ℓ denotes the uniform distribution over ℓ bits.

The GGM tree-based PRF construction [GGM84] from one-way functions yields a puncturable PRF where the punctured key sizes are polynomial in the size of the set S [BW13].

In this work we also make use of *injective* families of PRFs [SW14, Wat15]:

Definition 3. *A statistically injective (puncturable) PRF family with failure probability $\epsilon(\cdot)$ is a family of (puncturable) PRFs PRF, such that with probability $1-\epsilon(\lambda)$ over the random choice of key $k \leftarrow \mathsf{PRF.Gen}_F(1^\lambda)$, we have that $F(k,\cdot)$ is injective.*

If the failure probability function $\epsilon(\cdot)$ is not specified, then we assume that $\epsilon(\cdot)$ is a negligible function in the security parameter λ. Sahai and Waters [SW14] showed that assuming the existence of one-way functions there exists statistically injective puncturable PRF family with failure probability $2^{-\epsilon(\lambda)}$.

2.3 Indistinguishability Obfuscation

We recall the definition of indistinguishability obfuscation.

Definition 4 (Indistinguishability Obfuscator [GGH+13]). *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:*

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

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

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*

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

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 $0 < \varepsilon < 1$.

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 5. *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* $\perp$*.*

$\mathsf{Dec}(sk_{f,t'}, C_t/UC_t)$[11]: *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 IND-CUFE-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 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 CorTokGen and honest tokens via 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 HonUpdate oracles. Thereby, Enc' allows generating honest ciphertexts (under mpk) and HonUpdate allows updating ciphertexts which have been honestly[12] generated via Enc'

[11] 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.

[12] We require honestly generated ciphertexts as input to HonUpdate which we track in the model. This is reminiscent of Cohen's work [Coh19] which details the necessities of such a restriction.

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.

$\mathsf{IND\text{-}CUFE\text{-}CPA}$ security. We say that a CUFE scheme is $\mathsf{IND\text{-}CUFE\text{-}CPA}$-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 1 depicts the experiment.

Definition 6 ($\mathsf{IND\text{-}CUFE\text{-}CPA}$ security). *A CUFE scheme* CUFE *is* $\mathsf{IND\text{-}CUFE\text{-}CPA}$*-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 1.*

Remark 2. We model the experiment semi-adaptive (i.e., the target tag and messages are chosen by the adversary before it has access to oracles, but after it has seen the main public key) as well as adaptive (i.e., the adversary has access to the oracles before specifying target tag and messages). Note that in Figure 1, we cover this by setting $\mathcal{O}_1 = \bot$, i.e., the adversary has no access to oracles in the first phase, or $\mathcal{O}_1 = \mathcal{O}$, i.e., the adversary has access to the oracles throughout both phases, respectively. We note that it would also be possible to define the experiment in a weaker selective setting or either choosing only the tag or the messages in a semi-adaptive sense. This is straightforward to model and we omit it for the sake of simplicity. Moreover, we note that to move from a selective to an adaptive setting, one can utilize 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 semi-adaptive $\mathsf{IND\text{-}CUFE\text{-}CPA}$ 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 supports any predicate that can be represented as a circuit of arbitrary polynomial size. Moreover, we conjecture that one can obtain adaptive FE security either using the black-box transformation of Ananth et al. [ABSV15] along with applying complexity leveraging over the tag space or by directly extending the adaptively secure FE construction of Waters [Wat15] to the CUFE setting.

Experiment $\mathsf{Exp}^{\mathsf{ind\text{-}cufe\text{-}cpa}}_{\mathsf{CUFE},A}(\lambda, \mathcal{F})$
$(mpk, msk) \leftarrow \mathsf{Setup}(\lambda, \mathcal{F})$
$\mathcal{K} := \emptyset, \mathcal{C} := \mathcal{UC} := \emptyset, \mathcal{HT} := \mathcal{CT} := \emptyset, \mathsf{c} := \mathsf{uc} := 1, \mathsf{ht} := \mathsf{ct} := 1$
$(t^*, x_0^*, x_1^*, \mathsf{st}) \leftarrow A^{\mathcal{O}_1}(mpk)$
$b \leftarrow \{0,1\}$
$C^* \leftarrow \mathsf{Enc}(mpk, x_b, t^*)$
$\mathcal{C} := \mathcal{C} \cup \{(0, C^*, t^*)\}$
$b' \leftarrow A^{\mathcal{O}}(C^*, \mathsf{st})$
if A is not valid, then return $b'' \leftarrow \{0,1\}$
if $b' = b$, 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. 1. The IND-CUFE-CPA security notion for CUFE. If $\mathcal{O}_1 = \perp$, then we call the experiment semi-adaptive and if $\mathcal{O}_1 = \mathcal{O}$, then we call it adaptive. If $\mathcal{O}_1 = \perp$ and mpk is not initially given to A, then we call the experiment selective.

4.1 Puncturable Tag-Based Deterministic Encryption

Our generic construction relies on a primitive called puncturable tag-based deterministic encryption (PTDE), which can be seen as a tag-based variant of puncturable deterministic encryption (PDE) introduced by Waters [Wat15].

Definition 7 (Puncturable Tag-Based Deterministic Encryption). *A puncturable tag-based deterministic encryption (PTDE) scheme Σ with message space $\mathcal{M}$ and tag space $\mathcal{T}$, consists of the following algorithms: (possibly) randomized algorithms* Setup *and* Puncture*, along with deterministic algorithms* Enc *and* Dec.
- $\mathsf{Setup}(1^\lambda)$*, on input a security parameter 1^λ, outputs a key k.*
- $\mathsf{Enc}(k, t, m)$*, on input a key k, a tag $t \in \mathcal{T}$ and a message m, outputs a ciphertext c.*
- $\mathsf{Dec}(k, t, c)$*, on input a key k, a tag $t \in \mathcal{T}$ and a ciphertext c, outputs a message $m \in \mathcal{M} \cup \{\perp\}$.*
- $\mathsf{Puncture}(k, t, m_0, m_1)$*, on input a key k, a tag $t \in \mathcal{T}$ and a pair of messages $m_0, m_1 \in \mathcal{M}$, outputs a new key k^{t, m_0, m_1} (the superscript is used to indicate the tag and messages where the key is punctured).*

Correctness. We say that a PTDE scheme Σ is correct if there exists a negligible function negl, such that for all $\lambda \in \mathbb{N}$, for all $t \in \mathcal{T}$, for all pairs of messages $m_0, m_1 \in \mathcal{M}$, for all $k \leftarrow \Sigma.\mathsf{Setup}(1^\lambda)$ and

$k^{t,m_0,m_1} \leftarrow \Sigma.\mathsf{Puncture}(k, t, m_0, m_1)$, for all $m \neq m_0, m_1$, it holds that

$$\Pr\left[\Sigma.\mathsf{Dec}(k^{t,m_0,m_1}, \Sigma.\mathsf{Enc}(k,t,m)) \neq m\right] = \mathsf{negl}(\lambda).$$

Moreover, we have that for all m (including m_0, m_1),

$$\Pr\left[\Sigma.\mathsf{Dec}(k, \Sigma.\mathsf{Enc}(k,t,m)) \neq m\right] = \mathsf{negl}(\lambda).$$

Next, we recall the notion of (selective) indistinguishability security for PTDE.

Definition 8 (Indistinguishability Security for PTDE). *A PTDE scheme Σ is indistinguishability secure, if for all PPT adversaries A it holds that*

$$\mathsf{Adv}_{\Sigma,A}(\lambda) := \left| \Pr\left[\begin{array}{l} k \leftarrow \Sigma.\mathsf{Setup}(1^\lambda), \\ (t, m_0, m_1) \leftarrow A(1^\lambda), \\ k^{t,m_0,m_1} \leftarrow \Sigma.\mathsf{Puncture}(k, t, m_0, m_1) \\ b \leftarrow \{0,1\} \\ c_0 \leftarrow \Sigma.\mathsf{Enc}(k,t,m_b), c_1 \leftarrow \Sigma.\mathsf{Enc}(k,t,m_{1-b}) \\ b^* \leftarrow A(k^{t,m_0,m_1}, c_0, c_1): \\ b = b^* \end{array}\right] - \frac{1}{2} \right|,$$

is negligible.

Remark 3. Our definition allows for a key to be punctured at two messages and a tag, which extends the original PDE definition given in [Wat15] with a tag puncturing. We note that this differs from puncturable tag-based encryption given by Chvojka et al. [CJK20], which allows puncturing only at tags instead and constitutes a randomized encryption scheme.

Construction of PTDE. We extend the PDE construction given by Waters [Wat15] to additionally consider tags. Our PTDE scheme has message space $\mathcal{M} = \{0,1\}^\lambda$ and tag space $\mathcal{T} = \{0,1\}^\ell$. We make use of two (puncturable) PRF families, where the first one is an injective puncturable PRF F_1 that takes inputs from λ bits to $\ell = \ell(\lambda)$ bits, and the second one F_2 takes inputs from ℓ bits to λ bits. The construction is as follows.

- $\mathsf{Setup}(1^\lambda)$: Sample keys $k_1 \leftarrow \mathsf{PRF.Gen}_{F_1}(1^\lambda)$ and $k_2 \leftarrow \mathsf{PRF.Gen}_{F_2}(1^\lambda)$, and output $k := (k_1, k_2)$.
- $\mathsf{Enc}(k := (k_1, k_2), t, m)$: Deterministically compute and output a ciphertext

$$c := (c_1 = F_1(k_1, m), c_2 = F_2(k_2, c_1 \oplus t) \oplus m).$$

- $\mathsf{Dec}(k := (k_1, k_2), t, c := (c_1, c_2))$: Compute $m' = F_2(k_2, c_1 \oplus t) \oplus c_2$. If $F_1(k_1, m') = c_1$, then output m', otherwise output $\perp$.
- $\mathsf{Puncture}(k := (k_1, k_2), t, m_0, m_1)$: Compute $d = F_1(k_1, m_0)$ and $e = F_1(k_1, m_1)$. Compute $k_1^{m_0,m_1} \leftarrow \mathsf{PRF.Puncture}_{F_1}(k_1, \{m_0, m_1\})$ and $k_2^t \leftarrow \mathsf{PRF.Puncture}_{F_2}(k_2, \{d \oplus t, e \oplus t\})$, and output $k^{t,m_0,m_1} := (k_1^{m_0,m_1}, k_2^t)$.

The correctness for the non-punctured keys follows by observation, and correctness for key k^{t,m_0,m_1} on all messages $m \neq m_0, m_1$ holds as long as $F_1(k_1, m) \neq F_1(k_1, m_0)$ or $F_1(k_1, m_1)$, which holds because F_1 is injective. The security follows straightforwardly from the (punctured) PRF security of F_1 and F_2 and is established with the following theorem.

Theorem 1. *Let F_1 and F_2 be secure puncturable pseudorandom functions. Then, our construction is (selectively) indistinguishable secure PTDE scheme.*

Proof. The security proof follows via a sequence of hybrid games. Hereafter, let $\mathsf{Game}_i \approx \mathsf{Game}_{i+1}$ denote $|\Pr[\mathsf{Game}_i = 1] - \Pr[\mathsf{Game}_{i+1} = 1]| \leq \mathsf{negl}(\lambda)$.

Game_0: This corresponds to the honest execution of the (selective) indistinguishability game of PTDE.

Game_1: This is identical to Game_0 with the exception that the challenger randomly chooses c_1^b, c_1^{1-b} (when computing the challenge ciphertext) instead of computing $c_1^b = F_1(k_1, m_b)$ and $c_1^{1-b} = F_1(k_1, m_{1-b})$.
Game_2: This is identical to Game_1 with the exception that the challenger randomly chooses c_2^b, c_2^{1-b} (when computing the challenge ciphertext) instead of computing $c_2^b = F_2(k_2, c_1^b \oplus t)$ and $c_2^{1-b} = F_2(k_2, c_2^{1-b} \oplus t)$.

Lemma 1. *If F_1 is a selectively secure puncturable PRF, then it holds that $\mathsf{Game}_0 \approx \mathsf{Game}_1$.*

Proof. We describe a PPT reduction algorithm B that plays the selective puncturable tag-based deterministic encryption (PTDE) security game. B receives (t, m_0, m_1) from A and proceeds as in Game_0, except that it samples a bit $b \in \{0,1\}$, submits m_b, m_{1-b} to the punctured PRF challenger. B receives back a punctured PRF key $k_{\mathsf{prf}}^{m_b, m_{1-b}}$ and challenge values z_0, z_1. B sets $k_{\mathsf{ptde}} := (k_{\mathsf{prf}}^{m_b, m_{1-b}}, k_{\mathsf{prf},2})$, $c_0 := (z_b, F_2(k_{\mathsf{prf},2}, z_b \oplus t) \oplus m_b)$ and $c_1 := (z_{1-b}, F_2(k_{\mathsf{prf},2}, z_{1-b} \oplus t) \oplus m_{1-b})$ and returns $(k_{\mathsf{ptde}}, c_0, c_1)$ to A. If A wins, i.e., $b' = b$, then B outputs 1 to indicate that $z_0 = F_1(k_{\mathsf{prf},1}, m_0)$ and $z_1 = F_1(k_{\mathsf{prf},1}, m_1)$, for some PRF key $k_{\mathsf{prf},1}$, and otherwise, it outputs 0 to indicate that z_0, z_1 were random values.

We observe that if z_0, z_1 are generated as $z_0 = F_1(k_{\mathsf{prf},1}, m_0)$ and $z_1 = F_1(k_{\mathsf{prf},1}, m_1)$, then B gives the view of Game_0 to A. Otherwise, if z_0 and z_1 were chosen randomly, then the view is of Game_1. Therefore, if A can distinguish between the two games with non-negligible advantage, then B must also have non-negligible advantage against the puncturable PRF security game.

Lemma 2. *If F_2 is a selectively secure puncturable PRF, then it holds that $\mathsf{Game}_1 \approx \mathsf{Game}_2$.*

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

We note that since $c_1^b, c_1^{1-b}, c_2^b, c_2^{1-b}$ are all chosen uniformly at random in Game_2, we have that the challenge ciphertexts $c_b := (c_1^b, c_2^b)$, for $b \in \{0,1\}$, information-theoretically hide the bit b. This final information-theoretic argument depends on the fact that the distribution of $\mathsf{PRF.Puncture}_{F_1}(k_{\mathsf{prf},1}, \{m_0, m_1\})$ is the same as $\mathsf{PRF.Puncture}_{F_1}(k_{\mathsf{prf},1}, \{m_1, m_0\})$. This concludes the proof of Theorem 1. □

4.2 Generic CUFE from iO for any Function

The generic construction is inspired by the punctured programming approach to construct functional encryption from indistinguishability obfuscation, given by Waters [Wat15]. More precisely, the construction makes use of indistinguishability obfuscation $i\mathcal{O}$, puncturable tag-based deterministic encryption (PTDE) scheme Σ, puncturable pseudorandom function F and pseudorandom generator PRG. The construction is described below (where the parts in blue in programs PInit:2, PKey:2 and PUpdate:2 highlight the changes with respect to programs PInit:1, PKey:1 and PUpdate:1):

- $\mathsf{Setup}(1^\lambda, \mathcal{F})$: Compute the following:
  1. Sample $k_{\mathsf{prf},o} \leftarrow \mathsf{PRF.Gen}_F(1^\lambda)$ and $k_{\mathsf{prf},u} \leftarrow \mathsf{PRF.Gen}_F(1^\lambda)$.
  2. Compute an obfuscation $P_{pp} \leftarrow i\mathcal{O}(\text{PInit:1}[k_{\mathsf{prf},o}])$ for the program PInit:1$[k_{\mathsf{prf},o}]$[13].

  Output the main public/secret key pair $(mpk := P_{pp},\ msk := (k_{\mathsf{prf},o}, k_{\mathsf{prf},u}))$.
- $\mathsf{KeyGen}(msk := (k_{\mathsf{prf},o}, k_{\mathsf{prf},u}), f, t)$: Compute an obfuscation $P_{f,t} \leftarrow i\mathcal{O}(\text{PKey:1}[k_{\mathsf{prf},o}, k_{\mathsf{prf},u}, f, t])$ for the program PKey:1$[k_{\mathsf{prf},o}, k_{\mathsf{prf},u}, f, t]$[14]. Output the secret key $sk_{f,t} := P_{f,t}$.
- $\mathsf{TokGen}(msk := (k_{\mathsf{prf},o}, k_{\mathsf{prf}_u}), t, t')$: Compute an obfuscation $P_{t\to t'} \leftarrow i\mathcal{O}(\text{PUpdate:1}[k_{\mathsf{prf},o}, k_{\mathsf{prf},u}, t, t'])$ for the program PUpdate:1$[k_{\mathsf{prf},o}, k_{\mathsf{prf},u}, t, t']$[15]. Output the update token $\Delta_{t\to t'} := P_{t\to t'}$.
- $\mathsf{Enc}(mpk := P_{pp}, m, t)$: Compute the following:
  1. Sample a random $r \in \{0,1\}^\lambda$.
  2. Run the obfuscated program $(p, k_{\mathsf{ptde}}) \leftarrow P_{pp}(r)$.
  3. Compute $c \leftarrow \Sigma.\mathsf{Enc}(k_{\mathsf{ptde}}, t, m)$.

  Output the ciphertext $C_t := (p, c)$.

[13] The program is padded to the size equal to $\max\{|\text{PInit:1}[k_{\mathsf{prf},o}]|, |\text{PInit:2}[k_{\mathsf{prf},o}^{p^*}]|\}$.

[14] The program is padded to the size equal to $\max\{|\text{PKey:1}[k_{\mathsf{prf},o}, k_{\mathsf{prf},u}, f, t]|, |\text{PKey:2}[k_{\mathsf{prf},o}^{p^*}, k_{\mathsf{prf},u}, f, t, p^*, c_0, c_1, v_f, k'_{\mathsf{ptde}}]|\}$.

[15] The program is padded to the size equal to $\max\{|\text{PUpdate:1}[k_{\mathsf{prf},o}, k_{\mathsf{prf},u}, t, t']|, |\text{PUpdate:2}[k_{\mathsf{prf},o}^{p^*}, k_{\mathsf{prf},u}, t, t', p^*, c_0, c_1, c'_0, c'_1, k'_{\mathsf{ptde}}]|\}$.

PInit:1

Constants: PRF key $k_{\mathsf{prf},o}$
Inputs: $r \in \{0,1\}^\lambda$

1. Compute $p = \mathsf{PRG}(r)$.
2. Compute $k_{\mathsf{ptde}} = F(k_{\mathsf{prf},o}, p)$.
3. Output (p, k_{ptde}).

PInit:2

Constants: Punctured PRF key $k^{p^*}_{\mathsf{prf},o}$
Inputs: $r \in \{0,1\}^\lambda$

1. Compute $p = \mathsf{PRG}(r)$.
2. Compute $k_{\mathsf{ptde}} = F(k^{p^*}_{\mathsf{prf},o}, p)$.
3. Output (p, k).

PKey:1

Constants: PRF keys $k_{\mathsf{prf},o}, k_{\mathsf{prf},u}$, function description $f \in \mathcal{F}$, tag t
Inputs: $C_t := (p, c)$

1. If C_t is updated ciphertext, set $k_{\mathsf{prf}} := k_{\mathsf{prf},u}$, else set $k_{\mathsf{prf}} := k_{\mathsf{prf},o}$.
2. Compute $k_{\mathsf{ptde}} = F(k_{\mathsf{prf}}, p)$.
3. Output $f(\Sigma.\mathsf{Dec}(k_{\mathsf{ptde}}, t, c))$.

PKey:2

Constants: (Punctured) PRF keys $k^{p^*}_{\mathsf{prf},o}$, $k_{\mathsf{prf},u}$, function description $f \in \mathcal{F}$, tag t, point $p^* \in \{0,1\}^{2\lambda}$, PTDE ciphertexts c_0, c_1, value v_f, punctured PTDE key $k^*_{\mathsf{ptde}} := k^{t^*, m^*_0, m^*_1}_{\mathsf{ptde}}$
Inputs: $C_t := (p, c)$

1. If $p = p^*$ and $c \neq c_0, c_1$, output $f(\Sigma.\mathsf{Dec}(k^*_{\mathsf{ptde}}, t, c))$.
2. If $p = p^*$ and ($c = c_0$ or $c = c_1$), output v_f.
3. If C_t is updated ciphertext, set $k_{\mathsf{prf}} := k_{\mathsf{prf},u}$, else set $k_{\mathsf{prf}} := k^{p^*}_{\mathsf{prf},o}$.
4. Otherwise compute $k_{\mathsf{ptde}} = F(k_{\mathsf{prf}}, p)$ and output $f(\Sigma.\mathsf{Dec}(k_{\mathsf{ptde}}, t, c))$.

– $\mathsf{Update}(\Delta_{t\to t'} := P_{t\to t'}, C_t)$: Compute the following:
  1. Sample a random $r \in \{0,1\}^\lambda$.
  2. Run the obfuscated program $C_{t'} \leftarrow P_{t\to t'}(C_t, r)$.

  Output the updated ciphertext $C_{t'}$[16].
– $\mathsf{Dec}(sk_{f,t} := P_{f,t}, C_t)$: 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 puncturable tag-based deterministic encryption scheme Σ, puncturable pseudorandom function F, pseudorandom generator PRG, obfuscator $i\mathcal{O}$, and the description of the programs PInit:1, PKey:1 and PUpdate:1.

Next, we present the proof of $\mathsf{IND\text{-}CUFE\text{-}CPA}$ security of our generic construction.

Theorem 2. *Let Σ be a puncturable tag-based deterministic encryption scheme, F be a secure puncturable pseudorandom function, PRG be a secure pseudorandom generator, $i\mathcal{O}$ be an indistinguishability obfuscator*

[16] We assume that it is easy to distinguish between updated and fresh ciphertext. This is without loss of generality as we can simply append a bit to the ciphertexts to achieve this distinguishability.

PUpdate:1

Constants: PRF keys $k_{\mathsf{prf},o}, k_{\mathsf{prf},u}$, tags $t, t' \in \mathcal{T}$
Inputs: $C_t := (p, c), r \in \{0,1\}^\lambda$

1. Compute $k_{\mathsf{ptde}} \leftarrow F(k_{\mathsf{prf},o}, p)$.
2. Compute $m \leftarrow \Sigma.\mathsf{Dec}(k_{\mathsf{ptde}}, t, c)$.
3. Compute $p' \leftarrow \mathsf{PRG}(r)$.
4. Compute $k'_{\mathsf{ptde}} \leftarrow F(k_{\mathsf{prf},u}, p')$.
5. Compute $c' \leftarrow \Sigma.\mathsf{Enc}(k'_{\mathsf{ptde}}, t', m)$.
6. Output $C_{t'} := (p', c')$.

PUpdate:2

Constants: (Punctured) PRF keys $k^{p^*}_{\mathsf{prf},o}$, $k_{\mathsf{prf},u}$, tags $t, t' \in \mathcal{T}$, point $p^* \in \{0,1\}^{2\lambda}$, PTDE ciphertexts c_0, c_1, c'_0, c'_1, punctured PTDE key $k^*_{\mathsf{ptde}} := k^{t^*, m^*_0, m^*_1}_{\mathsf{ptde}}$
Inputs: $C_t := (p, c), r \in \{0,1\}^\lambda$

1. If $p = p^*$ and $c = c_b$, output $C_{t'} := (p, c'_b)$.
2. If $p = p^*$ and $c \neq c_0, c_1$
   - Compute $m \leftarrow \Sigma.\mathsf{Dec}(k^*_{\mathsf{ptde}}, t, c)$.
3. Else
   - Compute $k_{\mathsf{ptde}} = F(k^{p^*}_{\mathsf{prf},o}, p)$.
   - Compute $m \leftarrow \Sigma.\mathsf{Dec}(k_{\mathsf{ptde}}, t, c)$.
4. Compute $p' \leftarrow \mathsf{PRG}(r)$.
5. Compute $k'_{\mathsf{ptde}} \leftarrow F(k_{\mathsf{prf},u}, p')$
6. Compute $c' \leftarrow \Sigma.\mathsf{Enc}(k'_{\mathsf{ptde}}, t', m)$.
7. Output $C_{t'} := (p', c')$.

for the circuit class $\mathcal{C}_\lambda$. Then, our generic construction is a semi-adaptively IND-CUFE-CPA *secure CUFE scheme.*

Proof. The proof is organized in a sequence of hybrid games, where initially the challenger encrypts m_b for a random bit $b \in \{0,1\}$, and we gradually (in multiple steps) change the encryption into an encryption of m_0, which is independent of the bit b. We first define the sequence of games, and then, show (based on the security of different primitives) that any PPT adversary's advantage in each game must be negligibly close to the previous game. Hereafter, let $\mathsf{Game}_i \approx \mathsf{Game}_{i+1}$ denote $|\Pr[\mathsf{Game}_i = 1] - \Pr[\mathsf{Game}_{i+1} = 1]| \leq \mathsf{negl}(\lambda)$.

- Game_0: This corresponds to the honest execution of the semi-adaptive variant of the indistinguishability game given in Section 3. More precisely, the adversary is given the main public key mpk, then the adversary selects a challenge tag t^* and a challenge message pair m^*_0, m^*_1, and the challenger choses a bit $b \in \{0,1\}$ and encrypts m^*_b in the challenge ciphertext.
- Game_1: This is identical to Game_0 with the exception that the challenger choses a random $p^* \in \{0,1\}^{2\lambda}$ during the computation of the challenge ciphertext, instead of choosing a random $r^* \in \{0,1\}^\lambda$ and computing $p^* \leftarrow \mathsf{PRG}(r^*)$.
- Game_2: This is identical to Game_1 with the exception that the challenger computes the punctured key $k^{p^*}_{\mathsf{prf},o} \leftarrow \mathsf{PRF.Puncture}_F(k_{\mathsf{prf},o}, p^*)$ and sets $P_{pp} \leftarrow i\mathcal{O}(\text{PInit:2}[k^{p^*}_{\mathsf{prf},o}])$.
- Game_3: This is identical to Game_2 with the exception that for answering each secret key query $(f, t) \in (\mathcal{F} \times \mathcal{T})$, the challenger does the following: compute $k'_{\mathsf{ptde}} \leftarrow \Sigma.\mathsf{Puncture}(k^*_{\mathsf{ptde}}, t^*, m^*_0, m^*_1)$, for $k^*_{\mathsf{ptde}} \leftarrow F(k_{\mathsf{prf},o}, p^*)$, compute $c'_0 \leftarrow \Sigma.\mathsf{Enc}(k^*_{\mathsf{ptde}}, t, m^*_0)$, $c'_1 \leftarrow \Sigma.\mathsf{Enc}(k^*_{\mathsf{ptde}}, t, m^*_1)$, and $v_f = f(m^*_0) = f(m^*_1)$. Then, let c_0, c_1 consist of c'_0, c'_1 in lexicographic order[17], and the challenger responds with $P_{f,t} \leftarrow i\mathcal{O}(\text{PKey:2}[k^{p^*}_{\mathsf{prf},o}, k_{\mathsf{prf},u}, f, t, p^*, c_0, c_1, v_f, k'_{\mathsf{ptde}}])$.

[17] If $c'_0 < c'_1$, then $c_0 = c'_0, c_1 = c'_1$, otherwise, $c_0 = c'_1, c_1 = c'_0$.

– Game_4: This is identical to Game_3 with the exception that for answering each token generation query $(t, t') \in (\mathcal{T} \times \mathcal{T})$, the challenger does the following: compute $k'_{\mathsf{ptde}} \leftarrow \Sigma.\mathsf{Puncture}(k^*_{\mathsf{ptde}}, t^*, m^*_0, m^*_1)$, for $k^*_{\mathsf{ptde}} \leftarrow F(k_{\mathsf{prf},o}, p^*)$, compute $c_0 \leftarrow \Sigma.\mathsf{Enc}(k^*_{\mathsf{ptde}}, t, m^*_0)$, $c_1 \leftarrow \Sigma.\mathsf{Enc}(k^*_{\mathsf{ptde}}, t, m^*_1)$, and $c'_0 \leftarrow \Sigma.\mathsf{Enc}(k''_{\mathsf{ptde}}, t', m^*_0)$, $c'_1 \leftarrow \Sigma.\mathsf{Enc}(k''_{\mathsf{ptde}}, t', m^*_1)$, for $k''_{\mathsf{ptde}} \leftarrow F(k_{\mathsf{prf},u}, r)$ and random $r \in \{0,1\}^\lambda$. Then, sort and order c_0, c_1, c'_0, c'_1 lexicograpically[18] and respond with $P_{t \to t'} \leftarrow i\mathcal{O}(\text{PUpdate:2}[k^{p^*}_{\mathsf{prf},o}, k_{\mathsf{prf},u}, , t, t', p^*, c_0, c_1, c'_0, c'_1, k'_{\mathsf{ptde}}])$.
– Game_5: This is identical to Game_4 with the exception that the challenger samples a random k^*_{ptde} instead of computing it as $k^*_{\mathsf{ptde}} \leftarrow F(k_{\mathsf{prf},o}, p^*)$.
– Game_6: This is identical to Game_5 with the exception that the challenger encrypts m^*_0, i.e., the challenger computes $c^* \leftarrow \Sigma.\mathsf{Enc}(k^*_{\mathsf{ptde}}, t^*, m^*_0)$ and outputs (p^*, c^*).

Lemma 3. *If* PRG *is a secure pseudorandom generator, then it holds that* $\mathsf{Game}_0 \approx \mathsf{Game}_1$.

Proof. We describe a PPT reduction algorithm B that plays the PRG security game. First, B creates the main public/secret key pair (mpk, msk) (as in Game_0). Next, B receives a PRG challenge $p \in \{0,1\}^{2\lambda}$. Then, B runs the adversary A and executes the CUFE security game (as described in Game_0), with the exception that when computing the challenge ciphertext it sets $p^* := p$. We note that since B generates everything else itself (as in Game_0, it has all the necessary information to answer the oracle queries of A). Lastly, if A wins, i.e., $b' = b$, then B outputs 1 to indicate that p was in the image of PRG, and otherwise, it outputs 0 to indicate that p was chosen randomly.

We observe that if the PRG challenger generated $p = \mathsf{PRG}(r)$, for some $r \in \{0,1\}^\lambda$, then B gives the view of Game_0 to A. Otherwise, if p was chosen randomly, then the view is of Game_1. Therefore, if A can distinguish between the two games with non-negligible advantage, then B must also have non-negligible advantage against the PRG security game.

Lemma 4. *If* $i\mathcal{O}$ *is an indistinguishability obfuscator for the circuit class* $\mathcal{C}_\lambda$*, then it holds that* $\mathsf{Game}_1 \approx \mathsf{Game}_2$.

Proof. We construct a distinguisher B for $i\mathcal{O}$. B proceeds as in Game_1, with the exception that it computes the punctured PRF key $k^{p^*}_{\mathsf{prf},o} \leftarrow \mathsf{PRF.Puncture}_F(k_{\mathsf{prf},o}, p^*)$ and generates the two circuits $C_0 = \text{PInit:1}[k_{\mathsf{prf},o}]$ and $C_1 = \text{PInit:2}[k^{p^*}_{\mathsf{prf},o}]$. B submits C_0, C_1 to the $i\mathcal{O}$ challenger and receives back a program P, which it sets as $mpk := P_{pp} := P$, and returns it to the CUFE adversary A. The rest of the execution is identical to Game_1. If A wins, i.e., $b' = b$, then B outputs 0 to indicate that P was was an obfuscation of C_0, and otherwise, it outputs 1 to indicate that P was an obfuscation of C_1.

We observe that if the $i\mathcal{O}$ challenger generated P as an obfuscation of C_0, then B gives the view of Game_1 to A. Otherwise, if P was generated as an obfuscation of C_1, then the view is that of Game_2. Moreover, the programs are functionally equivalent with all but negligible probability, because p^* lies outside the image of PRG with probability at least $1 - 2^\lambda$. Therefore, if A can distinguish between the two games with non-negligible advantage, then B must also have non-negligible advantage against the $i\mathcal{O}$ security game for the circuit class $\mathcal{C}_\lambda$.

Lemma 5. *If* $i\mathcal{O}$ *is an indistinguishability obfuscator for the circuit class* $\mathcal{C}_\lambda$*, then it holds that* $\mathsf{Game}_2 \approx \mathsf{Game}_3$.

Proof. To prove this lemma we consider a hybrid argument. Let $Q_k = Q_k(\lambda)$ denote the number of secret key queries issued by the CUFE adversary A. For $i \in [0, Q_k]$ we define $\mathsf{Game}_{2,i}$ to be equivalent to Game_2 with the exception that the first i secret key queries are handled as in Game_3 and the last $Q_k - i$ are handled as in Game_2. Note that $\mathsf{Game}_{2,0}$ is the same as Game_2 and Game_{2,Q_k} is the same as Game_3. Hence, to prove security we need to establish that no adversary can distinguish between $\mathsf{Game}_{2,i}$ and $\mathsf{Game}_{2,i+1}$, for $i \in [0, Q_k - 1]$, with non-negligible advantage.

[18] Here we sort c_0 and c_1 lexicographically and then we order c'_0 and c'_1 according to this sort, i.e., if $c_0 \leftarrow \Sigma.\mathsf{Enc}(k_{\mathsf{ptde}}, t, m_b)$, then $c'_0 \leftarrow \Sigma.\mathsf{Enc}(k'_{\mathsf{ptde}}, t', m_b)$.

We construct a distinguisher B for $i\mathcal{O}$. B proceeds as in Game_2, except that the first i secret key queries are answered as in Game3. For query $i+1$, B computes $k'_{\mathsf{ptde}} \leftarrow \Sigma.\mathsf{Puncture}(k^*_{\mathsf{ptde}}, t^*, m^*_0, m^*_1)$, for $k^*_{\mathsf{ptde}} \leftarrow F(k_{\mathsf{prf},o}, p^*)$, computes $c'_0 \leftarrow \Sigma.\mathsf{Enc}(k^*_{\mathsf{ptde}}, t, m^*_0)$, $c'_1 \leftarrow \Sigma.\mathsf{Enc}(k^*_{\mathsf{ptde}}, t, m^*_1)$, and $v_f = f(m^*_0) = f(m^*_1)$, where f and t are the queried function and tag, respectively. Then, let c_0, c_1 consist of c'_0, c'_1 in lexicographic order, B generates the two circuits $C_0 = \text{PKey:1}[k_{\mathsf{prf},o}, k_{\mathsf{prf},u}, f, t]$ and $C_1 = \text{PKey:2}[k^{p^*}_{\mathsf{prf},o}, k_{\mathsf{prf},u}, f, t, p^*, c_0, c_1, v_f, k'_{\mathsf{ptde}}]$. B submits C_0, C_1 to the $i\mathcal{O}$ challenger and receives back a program P, which it sets as $sk_{f,t} := P_{f,t} := P$, and returns it to the CUFE adversary A as the query answer. If A wins, i.e., $b' = b$, then B outputs 0 to indicate that P was was an obfuscation of C_0, and otherwise, it outputs 1 to indicate that P was an obfuscation of C_1.

We observe that if the $i\mathcal{O}$ challenger generated P as an obfuscation of C_0, then B gives the view of $\mathsf{Game}_{2,i}$ to A. Otherwise, if P was generated as an obfuscation of C_1, then the view is that of $\mathsf{Game}_{2,i+1}$. Moreover, the programs are functionally equivalent with all but negligible probability, because the only difference in the programs is that the response is hardwired for the two inputs (i.e., for the challenge ciphertexts). Therefore, if A can distinguish between the two games with non-negligible advantage, then B must also have non-negligible advantage against the $i\mathcal{O}$ security game for the circuit class $\mathcal{C}_\lambda$.

Lemma 6. *If $i\mathcal{O}$ is an indistinguishability obfuscator for the circuit class $\mathcal{C}_\lambda$, then it holds that $\mathsf{Game}_3 \approx \mathsf{Game}_4$.*

Proof. To prove this lemma we consider a hybrid argument. Let Q_t denote the total number of token generation queries issued by the CUFE adversary A, where $Q_t = Q_{ht} + Q_{ct}$, such that $Q_{ht} = Q_{ht}(\lambda)$ and $Q_{ct} = Q_{ct}(\lambda)$ denote the number of honest and corrupted token generation queries, respectively. For $i \in [0, Q_t]$ we define $\mathsf{Game}_{3,i}$ to be equivalent to Game_3 with the exception that the first i token generation queries are handled as in Game_4 and the last $Q_t - i$ are handled as in Game_3. Note that $\mathsf{Game}_{3,0}$ is the same as Game_3 and Game_{3,Q_t} is the same as Game_4. Hence, to prove security we need to establish that no adversary can distinguish between $\mathsf{Game}_{3,i}$ and $\mathsf{Game}_{3,i+1}$, for $i \in [0, Q_t - 1]$, with non-negligible advantage.

We construct a distinguisher B for $i\mathcal{O}$. B proceeds as in Game_3, except that the first i token generation queries are answered as in Game_4. For query $i+1$, B computes $k'_{\mathsf{ptde}} \leftarrow \Sigma.\mathsf{Puncture}(k^*_{\mathsf{ptde}}, t^*, m^*_0, m^*_1)$, for $k^*_{\mathsf{ptde}} \leftarrow F(k_{\mathsf{prf},o}, p^*)$, computes $c_0 \leftarrow \Sigma.\mathsf{Enc}(k^*_{\mathsf{ptde}}, t, m^*_0)$, $c_1 \leftarrow \Sigma.\mathsf{Enc}(k^*_{\mathsf{ptde}}, t, m^*_1)$, and $c'_0 \leftarrow \Sigma.\mathsf{Enc}(k''_{\mathsf{ptde}}, t', m^*_0)$, $c'_1 \leftarrow \Sigma.\mathsf{Enc}(k''_{\mathsf{ptde}}, t', m^*_1)$, for $k''_{\mathsf{ptde}} \leftarrow F(k_{\mathsf{prf},u}, r)$ and random $r \in \{0,1\}^\lambda$, where t, t' are the queried tags. Then, B sorts and orders c_0, c_1, c'_0, c'_1 lexicograpically. B generates the two circuits $C_0 = \text{PUpdate:1}[k_{\mathsf{prf},o}, k_{\mathsf{prf},u}, t, t']$, and $C_1 = \text{PUpdate:2}[k^{p^*}_{\mathsf{prf},o}, k_{\mathsf{prf},u}, t, t', p^*, c_0, c_1, c'_0, c'_1, k'_{\mathsf{ptde}}]$. B submits C_0, C_1 to the $i\mathcal{O}$ challenger and receives back a program P, which it sets as $\Delta_{t\to t'} := P_{t\to t'} := P$. If the query was a corrupted token generation query, then B sends $\Delta_{t\to t'}$ to the CUFE adversary A as the query answer, and otherwise, it stores it locally. If A wins, i.e., $b' = b$, then B outputs 0 to indicate that P was was an obfuscation of C_0, and otherwise, it outputs 1 to indicate that P was an obfuscation of C_1.

We observe that if the $i\mathcal{O}$ challenger generated P as an obfuscation of C_0, then B gives the view of $\mathsf{Game}_{3,i}$ to A. Otherwise, if P was generated as an obfuscation of C_1, then the view is of $\mathsf{Game}_{3,i+1}$. Moreover, the programs are functionally equivalent with all but negligible probability, because the only difference in the programs is that the response is hardwired for the two inputs (i.e., for the challenge ciphertexts). Therefore, if A can distinguish between the two games with non-negligible advantage, then B must also have non-negligible advantage against the $i\mathcal{O}$ security game for the circuit class $\mathcal{C}_\lambda$.

Lemma 7. *If F is a selectively secure puncturable PRF, then it holds that $\mathsf{Game}_4 \approx \mathsf{Game}_5$.*

Proof. We describe a PPT reduction algorithm B that plays the selective puncturable PRF security game. B proceeds as in Game_4 in its interaction with the CUFE adversary A, except that it chooses a random $p^* \in \{0,1\}^{2\lambda}$ and submits it to the punctured PRF challenger. B receives back a punctured PRF key $k^{p^*}_{\mathsf{prf}}$ and a challenge value z. B sets $k^*_{\mathsf{ptde}} := z$ and uses the punctured PRF key $k^{p^*}_{\mathsf{prf}}$ to compute the challenge ciphertext and answer the oracle queries of A as in Game_4. If A wins, i.e., $b' = b$, then B outputs 1 to indicate that $z = F(k_{\mathsf{prf}}, p^*)$, for some PRF key k_{prf}, and otherwise, it outputs 0 to indicate that z was a random value.

We observe that if z is generated as $F(k_{\mathsf{prf}}, p^*)$, then B gives the view of Game_4 to A. Otherwise, if z was chosen randomly, then the view is that of Game_5. Therefore, if A can distinguish between the two games with non-negligible advantage, then B must also have non-negligible advantage against the puncturable PRF security game.

Lemma 8. *If Σ is a selectively secure puncturable tag-based deterministic encryption scheme, then it holds that $\mathsf{Game}_5 \approx \mathsf{Game}_6$.*

Proof. We note that the only difference between Game_5 and Game_6 is that in Game_6 the CUFE challenger always encrypts m_0^*, whereas in Game_5 the encrypted message could be m_0^* or m_1^*, depending on the coin flip b. Moreover, when $b = 0$, the views of these two games are identical. Hence, if there is any difference in adversary A's advantage in guessing b between Game_5 and Game_6 it must be solely conditioned on $b = 1$.

We describe a PPT reduction algorithm B that plays the selective puncturable tag-based deterministic encryption (PTDE) security game. B proceeds as in Game_5, except that it submits the challenge messages m_0^*, m_1^* and tag t^* (given by A) to the PTDE challenger, which replies with a punctured PTDE key $k'_{\mathsf{ptde}} \leftarrow \Sigma.\mathsf{Puncture}(k^*_{\mathsf{ptde}}, t^*, m_0^*, m_1^*)$ and two ciphertexts c'_0, c'_1. B sets the challenge CUFE ciphertext to $C_{t^*} := (p^*, c^* := c'_0)$.

Let c_0, c_1 consist of c'_0, c'_1 in lexicographic order. Then, for answering each secret key query (of the form (f, t)), B computes $v_f = f(m_0^*) = f(m_1^*)$, and uses the punctured PTDE key k'_{ptde} to construct $P_{f,t} := \mathrm{PKey}{:}2[k^{p^*}_{\mathsf{prf},o}, k_{\mathsf{prf},u}, f, t, p^*, c_0, c_1, v_f, k'_{\mathsf{ptde}}]$. Similarly, for answering each token generation query (of the form (t, t')), B guesses a $\gamma \in \{0, 1\}$, computes $c''_\gamma \leftarrow \Sigma.\mathsf{Enc}(k''_{\mathsf{ptde}}, t', m^*_\gamma)$, $c'_{1-\gamma} \leftarrow \Sigma.\mathsf{Enc}(k''_{\mathsf{ptde}}, t', m^*_{1-\gamma})$, for $k''_{\mathsf{ptde}} \leftarrow F(k_{\mathsf{prf},u}, r)$ and random $r \in \{0,1\}^\lambda$. Then, B uses the previously computed values and the punctured PTDE key k'_{ptde} to construct $P_{t\to t'} := \mathrm{PUpdate}{:}2[k^{p^*}_{\mathsf{prf},o}, k_{\mathsf{prf},u}, t, t', p^*, c_0, c_1, c''_0, c''_1, k'_{\mathsf{ptde}}]$ and answer the token generation query. We note here that the guessed γ incurs a $1/2$ security loss. Encryption queries are answered in a straightforward way using the program $\mathrm{PInit}{:}2[k^{p^*}_{\mathsf{prf},o}]$.

Lastly, if A wins, i.e., $b' = b$, then B outputs 1 to indicate that $c^* := c'_0$ was an encryption of m_1^*, and otherwise, it outputs 0 to indicate that $c^* := c'_0$ was an encryption of m_0^*.

We observe that if $c^* := c'_0$ is generated as $\Sigma.\mathsf{Enc}(k^*_{\mathsf{ptde}}, m_1)$, then B gives the view of Game_5 (conditioned on $b = 1$) to A. Otherwise, if $c^* := c'_0$ is generated as $\Sigma.\mathsf{Enc}(k^*_{\mathsf{ptde}}, m_0)$, then the view is of Game_6. Therefore, if A can distinguish between the two games with non-negligible advantage, then B must also have non-negligible advantage against the puncturable tag-based deterministic encryption security game.

This concludes the proof of Theorem 2. □

4.3 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 tags for the PTDE scheme 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.2.

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 $\Lambda_q^\perp(\mathbf{A}) := \{\vec{u} \in \mathbb{Z}^m : \mathbf{A}\vec{u} = \vec{0} \bmod q\}$.

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* TrapGen, SampleD, SamplePre *such that:*

1. 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. 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 model (ROM), of a lattice-based identity-based IPFE scheme, and proved their security[19] under the $\mathrm{LWE}_{q,\alpha,n}$ assumption (Definition 10). Their constructions are in turn based on the IPFE scheme of Agrawal et al. [ALS16], ALS, described in Figure 2.

In our construction, we start from the ROM 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,

$\underline{\mathsf{Setup}(1^\lambda, n)}$:
$\mathbf{A} \leftarrow\!\!\$\ \mathbb{Z}_{q_{\mathsf{ALS}}}^{n\times m}$
$\mathbf{Z} \leftarrow\!\!\$\ D_{\mathbb{Z}^{m\times\ell},\rho_{\mathsf{ALS}}}$
$\mathbf{D} \leftarrow \mathbf{AZ}$
$\mathsf{mpk} \leftarrow (\mathbf{A}, \mathbf{D})$
$\mathsf{msk} \leftarrow \mathbf{Z}$
Return $(\mathsf{mpk}, \mathsf{msk})$

$\underline{\mathsf{KeyGen}(\mathbf{Z}, \vec{y} \in \mathcal{Y})}$:
Return $(\vec{y}, \mathsf{sk}_{\vec{y}} := \mathbf{Z} \cdot \vec{y})$

$\underline{\mathsf{Enc}(\mathsf{mpk}, \vec{x} \in \mathcal{X})}$:
$\vec{s} \leftarrow\!\!\$\ \mathbb{Z}_{q_{\mathsf{ALS}}}^{n}$
$\vec{e}_1 \leftarrow\!\!\$\ \mathcal{D}_{\mathbb{Z}^m,\sigma_{\mathsf{ALS}}}$
$\vec{e}_2 \leftarrow\!\!\$\ \mathcal{D}_{\mathbb{Z}^\ell,\sigma_{\mathsf{ALS}}}$
$\mathsf{ct}_1 = \mathbf{A}^\top \vec{s} + \vec{e}_1$
$\mathsf{ct}_2 = \mathbf{D}^\top \vec{s} + \vec{e}_2 + \left\lfloor \frac{q}{K} \right\rfloor \cdot \vec{x}$
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 $\arg\min_{\mu\in\{0,\ldots,K+1\}} \left| \left\lfloor \frac{q}{K} \right\rfloor \cdot \mu' - \mu \right|$

Fig. 2. Inner-product functional encryption scheme ALS, with parameters as in [ACGU20].

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

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

i.e., updating ciphertexts. Thus, even though our construction is only proved secure in the ROM, it also supports a richer class of functionalities than previous works. Our lattice-based CUFE construction is described in Figure 3. Dimensions of matrices involved in the construction are presented in Table 1.

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. 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

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

[19] We refer the reader to Appendix A.1 for a formal definition.

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

$\underline{\mathsf{KeyGen}((\mathbf{A}, \mathbf{T_A}), \vec{y} \in \mathcal{Y}, t)}$:
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 $(\vec{y}, \{\mathsf{sk}_{\vec{y},t,\ell} := \mathbf{Z}_{t,\ell} \cdot \vec{y}\}_{\ell=1,2})$

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

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

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

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

Fig. 3. Lattice-based Ciphertext-Updatable IPFE scheme.

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

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

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 $\vec{x} \in \{0, \ldots, P\}^m$ under a tag t. Secret keys corresponds to a tag t and a vector $\vec{y} \in \{0, \ldots, V\}^m$. When tags match, our scheme decrypts the bounded inner-product $\langle \vec{x}, \vec{y} \rangle \in \{0, \ldots, K\}$, 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. 3 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,[20]
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 ROM, 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], who show how to generically obtain proxy re-encryption (PRE) schemes secure against adversaries that can adaptively corrupt users from PRE schemes secure against adversaries that cannot make adaptive user corruptions. They do so by reducing the simulation in the security proof to a pebbling game on the graph "underlying" the security game [JKK+17]. We believe that any improvement to the results of Fuchsbauer et al. [FKKP19] and Jafargholi et al. [JKK+17] could also offer useful insights on how to overcome the current limitation of our construction.

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 3 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.

[20] observe that this limitation is only *per tag*

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 $i \in [Q_\mathrm{h}], i \neq i^*$, 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 $\mathbf{Y}_{t',t} \leftarrow \mathsf{SamplePre}(\mathbf{B}_{t_i,1}, \mathbf{T}_{\mathbf{B}_{t_i,1}}, \mathbf{B}_{t,2} - \mathbf{A}\mathbf{X}_{t_i,t}, \rho)$, along with $\mathbf{R}_{t_i\to t,2} \leftarrow \mathsf{SampleLeft}(\mathbf{B}_{t_i,1}, \mathbf{T}_{\mathbf{B}_{t_i,1}}, \mathbf{A}, \mathbf{D}_t - \mathbf{D}_{t_i}, \rho)$. 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 $i \neq i^*$):

– 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

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

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 $\mathbf{T_A}$ 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 $\{j_k\}_{k\leq Q_\mathrm{u}}$ (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 $\mathbf{S}_{t^*} \leftarrow \{\pm 1\}^{m\times m}$ 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 $\mathbf{S}_{t_{j_k}} \leftarrow \{\pm 1\}^{m\times m}$ 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 AD-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

$$sk_{t^*,1,\vec{y}} := \left(\frac{sk_{\vec{y}}}{\mathbf{Z}_{t^*}\vec{y}}\right),$$

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

$$sk_{t_{j_k},2,\vec{y}} := \left(\frac{sk_{\vec{y}} + \mathbf{R}_{t_{j_k}}\vec{y}}{\mathbf{Z}_{t^*}\vec{y}}\right),$$

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.2.)[21]

- 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.

In this game, the advantage of the adversary is upper bounded by the advantage of breaking the ALS scheme, i.e., that $\mathbf{Adv}_{\mathsf{Game}_8}(\mathcal{A}) \leq \mathbf{Adv}_{\mathrm{ALS}}(\mathcal{A})$. 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 $k \in [Q_{\mathrm{u}}]$, 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 $\mathbf{Z}_{\mathrm{ALS}}$, 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

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

□

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 indistinguishability obfuscation (iO) and is proven to achieve semi-adaptive 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 several questions as interesting open problems. Firstly, to construct a CUFE scheme that satisfies our adaptive security model without any further restrictions or bound on the number of oracle queries. Secondly, to construct practical CUFE schemes for a richer class of functionalities, e.g., quadratic functions, which can further broaden the scope of application. Thirdly, we consider it an interesting direction to study multi-input as well as multi-client extensions of CUFE similarly as it has been done for IB- and AB-IPFE in [ACGU20] and [NPP22], respectively.

[21] 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.

Acknowledgements. We want to thank the anonymous reviewers for their helpful comments and suggestions. In particular we want to thank one anonymous reviewer from the Journal of Cryptology to point out a problem with our initial generic CUFE construction. 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. The work of Valerio Cini and Daniel Slamanig was done while both were with AIT Austrian Institute of Technology.

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.

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.

AGW20. Michel Abdalla, Junqing Gong, and Hoeteck Wee. Functional encryption for attribute-weighted sums from k-Lin. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 685–716. Springer, Heidelberg, August 2020.

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.

BCS22. Élie Bouscatié, Guilhem Castagnos, and Olivier Sanders. Pattern matching in encrypted stream from inner product encryption. Cryptology ePrint Archive, Report 2022/1527, 2022. https://eprint.iacr.org/2022/1527.

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.

BW13. Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 280–300. Springer, Heidelberg, December 2013.

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.

CJK20. Peter Chvojka, Tibor Jager, and Saqib A. Kakvi. Offline witness encryption with semi-adaptive security. In Mauro Conti, Jianying Zhou, Emiliano Casalicchio, and Angelo Spognardi, editors, *ACNS 20, Part I*, volume 12146 of *LNCS*, pages 231–250. Springer, Heidelberg, October 2020.

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.

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.

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.

dPP22. Paola de Perthuis and David Pointcheval. Two-client inner-product functional encryption with an application to money-laundering detection. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 725–737. ACM Press, November 2022.

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.

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.

GGM84. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th FOCS*, pages 464–479. IEEE Computer Society Press, October 1984.

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.

JKK+17. Zahra Jafargholi, Chethan Kamath, Karen Klein, Ilan Komargodski, Krzysztof Pietrzak, and Daniel Wichs. Be adaptive, avoid overcommitting. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 133–163. Springer, Heidelberg, August 2017.

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.

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.

MP12. Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, Heidelberg, April 2012.

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. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part I*, volume 13791 of *LNCS*, pages 95–125. Springer, Heidelberg, December 2022.

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.

SW14. Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th ACM STOC*, pages 475–484. ACM Press, May / June 2014.

Wat15. Brent Waters. A punctured programming approach to adaptively secure functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 678–697. Springer, Heidelberg, August 2015.

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 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* $\mathcal{O}_{\mathsf{KeyGen}}(\cdot)$ *is an oracle that on*

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

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* AD-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.2 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 $\frac{2}{e^{\pi t^2}}$.*