Ciphertext-Updatable Attribute-Based and Predicate Encryption from Lattices

Robert Schädlich[1⋆], Linda Scheu-Hachtel[2], Erkan Tairi[3⋆], and Yuejun Wang[4]

[1] King's College London
robert.schaedlich@kcl.ac.uk

[2] University of Mannheim
linda.scheu-hachtel@uni-mannheim.de

[3] University of California, Berkeley
erkan.tairi@berkeley.edu

[4] State Key Laboratory of Integrated Services Networks, Xidian University, China
yuejun.w@stu.xidian.edu.cn

Abstract. We present a novel variant of attribute-based encryption (ABE) and predicate encryption (PE) which supports ciphertext updates, called ciphertext-updatable ABE (CU-ABE) and PE (CU-PE). Such a feature enhances the usability of the fine-grained encryption paradigm by allowing controlled updates to the ciphertexts. Updating ciphertexts is carried out via update tokens, which can only be generated by the master secret key holder, yet any party with access to the token can convert the ciphertexts.

Apart from defining these new primitives and establishing their security in the indistinguishability-based setting, we provide various constructions with different trade-offs. Concretely, we provide a generic transformation from plain ABE to unidirectional and single-hop CU-ABE using only (inner-product) functional encryption (IPFE), which can be instantiated from plain LWE. Then, we show how to combine this CU-ABE with lockable obfuscation to obtain unidirectional and single-hop CU-PE from LWE. These constructions support bounded number of update tokens.

While unidirectional and single-hop updates with bounded number of tokens are sufficient for practical applications, we show that we can extend our results to multi-hop and unbounded token setting by constructing both key-policy and ciphertext-policy CU-ABE schemes for all bounded-depth circuits. Proving security of these multi-hop constructions turned out to be non-trivial, which required us to develop novel techniques and rely on public-coin evasive LWE assumption.

1 Introduction

Attribute-based encryption (ABE) [SW05, GPSW06] is an extension of public-key encryption that supports fine-grained access to the encrypted data. In ciphertext-policy ABE (CP-ABE), the ciphertexts are associated with access policies, such as $f := ((\texttt{role} : \texttt{Nurse})\ \texttt{OR}\ (\texttt{role} : \texttt{Doctor}))\ \texttt{AND}\ (\texttt{dep} : \texttt{Cardiology})$, whereas the keys are associated with attributes like $\mathbf{x} := ((\texttt{role} : \texttt{Doctor}), (\texttt{dep} : \texttt{Cardiology}))$, and decryption is only possible when the attributes satisfy the access policy, i.e., $f(\mathbf{x}) = 0$.[1] On the other hand, key-policy ABE (KP-ABE) is the dual of CP-ABE, where the roles of the policies and attributes are switched, i.e., ciphertexts encode attributes and keys encode policies. A further strengthening of this notion, referred to as predicate encryption (PE), additionally enables hiding of the attributes, which are public in ABE.

There has been substantial progress in the past decade in the design and analysis of ABE and PE schemes, which led to numerous schemes with various trade-offs between efficiency, security and underlying assumptions. Moreover, ABE and PE have found use in variety of applications, e.g., electronic medical records

⋆ Work done while the authors were affiliated with ENS Paris.

[1] We follow here the convention that $f(\mathbf{x}) = 0$ signifies the ability to decrypt, which is a standard convention used in lattice-based ABE schemes.

and online social networks, and have been already deployed in practice, such as the Geo Key Manager of Cloudflare, which is used to distribute private key storage across data centers.[2] However, given our increasing dependence on trustworthy digital services, where we store an increasingly large amounts of sensitive data and necessity for complex access control requirements, we need ABE and PE schemes that can readily accommodate the change of roles, entities, attributes and policies. This implies that the deployed schemes should be flexible enough to allow quickly changing the access control associated with the ciphertexts.

Motivation for Ciphertext Updates. Let us consider the aforementioned Geo Key Manager application. Cloudflare stores its customers' TLS private keys, such that it can perform TLS handshake on their behalf in order to protect them against DDoS attacks or improve performance by caching. However, Cloudflare has numerous data centers distributed across the globe, which means that every machine in each of these data centers needs to have a local copy of the customers' TLS private keys. Though, a customer may demand that their private keys are only stored in specific regions, such as in EU data centers for example. In order to provide such a fine-grained access control, Cloudflare employed CP-ABE.

At a high level, the Geo Key Manager solution of Cloudflare works as follows: a customer uploads its TLS certificate and specifies an access policy associated with it, e.g., $f := (\texttt{region} : \texttt{EU})\ \texttt{OR}\ (\texttt{country} : \texttt{US})$. The central authority encrypts the customer's private key with the master public key under the specified policy and distributes the ciphertext across its data centers. Moreover, the central authority generates secret keys for the attributes associated to the data centers using its master secret key, and distributes these secret keys to the corresponding data centers. Next, when a user visits the customer's website, the TLS service at the data center that first receives the request fetches the ciphertext corresponding to the customer's private key, and decrypts it to obtain the private key and perform the TLS handshake on behalf of the customer. If the attributes of the data center do not satisfy the policy, then it forwards the request to the closest data center that has attributes satisfying the policy.

We now envision that in addition to having a fine-grained control, Cloudflare might also want to allow its customers to update the policies associated with their private keys. This would mean that Cloudflare would need to encrypt the data anew under this new policy and distribute it again to all the data centers. While this might be manageable if the data centers hold small number of ciphertexts per customer, this might be prohibitive if the number of ciphertexts per customer grows, and in this case we need a more efficient update mechanism. Hence, we consider an ABE scheme with a capability to update ciphertexts in a way that we can change the policy (or attributes in the case of KP-ABE) associated with it using some additional information (called an update token). To keep a fine-grained control over ciphertext updates we want to restrict that only the central authority can produce the update tokens necessary for ciphertext updates and consider only *unidirectional* updates[3], i.e., an update token $\Delta_{f \to f'}$ allows to update the ciphertext ct_f to $\mathsf{ct}_{f'}$ but not vice versa. Going back to our example, using such a ciphertext-updatable ABE (CU-ABE), the central authority of Cloudflare can produce update tokens whenever a customer wants to update the policy associated with its private keys, and distribute this update token to the data centers. Then, the data centers can update the policy associated to the ciphertexts of the customer using this update token. It is clear that such an update mechanism offers a more scalable solution than naively re-encrypting all the data (assuming the customer stores a large amount of data), since only a single update token needs to be transferred to the data centers.

We note that this functionality goes beyond what is provided by plain ABE schemes, ABE with switchable attributes (ABE-SA) schemes [DGP22, LWYW24], or attribute-based proxy re-encryption (AB-PRE) schemes [LCLS09, SDDR21, LAKWC21]. The plain ABE schemes are not updatable, i.e., they do not allow changing the policies or attributes associated with the ciphertexts, whereas ABE-SA and AB-PRE schemes either only allow delegation to a more restricted policy than the original one or allow the secret key holders to produce a switch or re-encryption key for arbitrary policies without the help of the central authority. Therefore, the existing models and schemes do not satisfy our fine-grained update requirement, which is necessary for the aforementioned application. More concretely, using an ABE-SA or AB-PRE scheme within

[2] https://blog.cloudflare.com/inside-geo-key-manager-v2/

[3] From practical standpoint, unidirectional and single-hop updates are preferable because they allow to keep a fine-grained control over the ciphertext updates.

Geo Key Manager would mean that if a data center gets compromised, then the attacker that has access to the secret key of the data center can produce re-encryption keys itself without the help of the central authority (even though it has limited power over what it can re-encrypt).

Additionally, even though the current deployment of Geo Key Manager uses a pairing-based ABE scheme [TKN20], given the recent standardization efforts of post-quantum cryptographic primitives by NIST[4], it is highly desirable to transition to post-quantum alternatives. Hence, apart from having fine-grained ciphertext updatability, we also want the underlying ABE to be post-quantum secure.

Since currently ABE schemes that achieve the desired properties are absent in the cryptographic literature, in this work we ask the following:

Can we define and construct (plausibly post-quantum secure) ciphertext-updatable[5] ABE (and PE)?

1.1 Our Contributions

We answer the above question affirmatively via our four-fold contribution:

1. We define ciphertext-updatable attribute-based encryption (CU-ABE) and predicate encryption (CU-PE) along with a security notion in the indistinguishability setting. To do so we extend the definition of plain ABE and PE with two additional algorithms, namely TokGen and Update (as done for functional encryption in [CRS+23]). The token generation algorithm allows the master secret key holder to produce an update token from an attribute (in the case of KP-ABE) or policy (in the case of CP-ABE) x into a new one x', i.e., $\Delta_{x\to x'} \leftarrow \mathsf{TokGen}(\mathsf{msk}, x, x')$. Furthermore, we allow the update token $\Delta_{x\to x'}$ to update the attribute/policy attached to a ciphertext ct_x by producing a new ciphertext $\mathsf{ct}_{x'}$, i.e., $\mathsf{ct}_{x'} \leftarrow \mathsf{Update}(\Delta_{x\to x'}, \mathsf{ct}_x)$. We note that we only consider *unidirectional* updates, but with both single-hop and multi-hop (in order to provide expressiveness), since unidirectional updates allow for finer control over the updates.
2. We give a generic transformation from plain ABE to CU-ABE using functional encryption (FE). The resulting scheme supports unidirectional *single-hop* update, supports bounded number of update tokens and inherits the security properties of the underlying plain ABE. We note that by considering ABEs with linearly constructible ciphertexts, i.e., where the ciphertexts are computed as inner-product of two vectors, we can restrict the requirement of FE to inner-product FE (IPFE) which can be instantiated from LWE. We also show that many existing ABEs have such linearly constructible ciphertexts. This allows us even to obtain CU-ABE for RAM circuits following the recent RAM-ABE construction given in [DHM+24].
3. We present a generic transformation from CU-ABE to ciphertext-updatable PE (CU-PE) using lockable obfuscation, which also can be instantiated from (plain) LWE. Our construction allows for unidirectional *single-hop* updates, supports bounded number of tokens and inherits the security guarantees of the underlying CU-ABE scheme. We leave it as an interesting open problem to extend our construction to multi-hop update setting.
4. We present constructions of (bounded-attribute) ciphertext-policy CU-ABE (CP-CU-ABE) and key-policy CU-ABE (KP-CU-ABE) for all bounded depth circuits in the standard model from *public-coin* evasive LWE assumption (which resists all known counterexamples [BÜW24, DJM+25, HJL25, AMYY25]). Notably, our constructions allow for unidirectional *multi-hop* updates, support *unbounded* number of tokens and they are plausibly post-quantum secure. We remark that while we can remove the bound on attribute length by applying the generic transformation of [GKW16], we leave it as an interesting open problem to remove the bounded depth restriction, since the recent results on bootstrapping lattice-based homomorphic evaluations [HLL23a] are not directly compatible with updating attribute encoding which we need in this work. We also leave it as an interesting theoretical question to prove the security

[4] https://csrc.nist.gov/projects/post-quantum-cryptography, https://csrc.nist.gov/projects/pqc-dig-sig

[5] Our naming of "ciphertext-updatable ABE" is derived from a recently given primitive *ciphertext-updatable functional encryption* [CRS+23], which offers similar features.

of our multi-hop constructions from *plain* LWE. Due to their reliance on evasive LWE, the concrete constructions and their ideas are presented in Section 5.

1.2 Related Work

Our ciphertext-updatable ABE is closely related to attribute-based proxy re-encryption (AB-PRE) [LCLS09, LFSW13, YJR+14, JSMG18, Fug21, SDDR21, LAKWC21, PO22]. In AB-PRE, which is an extension of plain PRE, a proxy can convert ciphertexts satisfying a policy function into ciphertexts satisfying a new policy function with the usage of a re-encryption key produced by the secret key holder, hence, the central authority (i.e., master secret key holder) is not involved for generating the re-encryption keys. We note that the transformation is always performed on a policy-level, independent of whether we are in the KP-ABE or CP-ABE setting. While nearly all existing AB-PRE constructions are not post-quantum secure, recently Susilo et al. [SDDR21] constructed the first plausibly post-quantum secure AB-PRE based on the LWE assumption. However, they only provided unidirectional single-hop key-policy AB-PRE construction, and this was recently extended to multi-hop setting by Luo et al. [LAKWC21].

Another line of related work to ours is ABE with switchable attributes, first introduced by Delerablée et al. [DGP22] and later extended by Luo et al. [LWYW24]. While both works target KP-ABE setting only, the work of Delerablée et al. [DGP22] merely allows delegation of secret keys to a more restricted policy than the original one, hence it does not allow any ciphertext updates, and moreover, it is not post-quantum secure. On the other hand, the work of Luo et al. [LWYW24] while being plausibly post-quantum secure (as it is based on the LWE assumption) and allowing ciphertext updates using so-called switch keys, its update mechanism is not fine-grained. Concretely, any secret key holder can generate such switch keys and can update the attributes associated to the ciphertexts without the help of the central authority, which is not suitable for our envision applications.

Revocable ABE schemes [LAKW+23, GWMM23, YCJ23] also deal with the scenario that the access structure might change. However, they are only able to take access permissions away, and not grant them. On the other hand, our ciphertext-updatable ABE constructions can be used to both revoke and grant access.

Another related work is ciphertext-updatable FE by Cini et al. [CRS+23]. Indeed, our security model (given in Section 3) is inspired by their model, however, their work is tailored to (IP)FE setting and is not suitable for our envisioned applications. Lastly, we note that one can achieve updatability of cryptographic primitives generically following the approach of Ananth et al. [ACJ17], however, their construction is non-black-box and uses updatable randomized encodings, which we only know how to construct from iO or FE for all circuits.

1.3 Technical Overview

In this section we present the high-level ideas of our constructions based on plain LWE. We first recall that a CU-ABE scheme equips a traditional ABE with two additional algorithms:

- Given the master secret key msk and a pair of ciphertext attributes (x, x'), the TokGen algorithm outputs an update token $\Delta_{x \to x'}$.
- Given an update token $\Delta_{x \to x'}$ and a ciphertext ct_x encrypted w.r.t. x, the Update algorithm outputs an updated ciphertext $\mathsf{ct}_{x'}$ which encrypts the same message as ct_x w.r.t. the attribute x'.

For security, we intuitively require that (1) a token allows updating the attributes of a ciphertext but not decrypting it, and (2) seeing an updated ciphertext is not sufficient to update other ciphertexts as well, i.e., an updated ciphertext does not disclose the token used during its update procedure. The formal security definition is provided in Definition 13, which is preceded by an informal explanation of the requirements (1) and (2).

Single-Hop CU-ABE from LWE. To begin with, let us consider an ABE scheme $\mathsf{ABE} = (\mathsf{aSetup}, \mathsf{aEnc}, \mathsf{aKeyGen}, \mathsf{aDec})$ and a FE scheme $\mathsf{iFE} = (\mathsf{iSetup}, \mathsf{iEnc}, \mathsf{iKeyGen}, \mathsf{iDec})$ for general circuits.

Basic Idea. To turn ABE into a CU-ABE, we can use iFE in a very naive manner:

- To encrypt a message μ under an attribute x, we run ABE encryption $\mathsf{aCT}_x \leftarrow \mathsf{aEnc}(\mathsf{aMPK}, x, \mu)$. For later usage, we also generate an iFE ciphertext $\mathsf{iCT}_x \leftarrow \mathsf{iEnc}(\mathsf{iMPK}, (\mu, r))$, where r is a fresh sample of random coins to be used by aEnc.
- Key generation is the same as in ABE.
- To generate an update token $\Delta_{x \to x'}$ for a pair of attributes (x, x'), we generate an iFE secret $\mathsf{iSK}_{x, C[\mathsf{aMPK}, x']}$ for a circuit $C[\mathsf{aMPK}, x'](\mu, r)$ which has the ABE public key aMPK and the attribute x' hardwired in its description, takes as input a message μ and random coins r, and evaluates the ABE encryption algorithm $\mathsf{aEnc}(\mathsf{aMPK}, x', \mu; r)$.
- To update a ciphertext $(\mathsf{aCT}_x, \mathsf{iCT}_x)$ using a token $\Delta_{x \to x'} = \mathsf{iSK}_{x, C[\mathsf{aMPK}, x']}$, we compute and output $\mathsf{aCT}_{x'} \leftarrow \mathsf{iDec}(\mathsf{iSK}_{x, C[\mathsf{aMPK}, x']}, \mathsf{iCT}_x)$.

This approach faces two immediate issues. Firstly, the update token $\Delta_{x \to x'} = \mathsf{iSK}_{x, C[\mathsf{aMPK}, x']}$ can be used to update any ciphertext $\mathsf{ct}_{x''}$ with attribute x'' to a ciphertext with attribute x', even if $x \neq x''$. Secondly, updating a ciphertext twice with different update tokens creates two ABE ciphertexts encrypted under the same randomness r. In this case, we lose the security guarantees of ABE. The first issue can be solved easily by using an identity-based FE scheme. More specifically, at the time of encryption, we run $\mathsf{iCT}_x \leftarrow \mathsf{iEnc}(\mathsf{iMPK}, x, (\mu, r))$, where x is the identity with respect to which we generate the ciphertext. Correspondingly, to generate a token $\Delta_{x \to x'}$, we compute $\mathsf{iSK}_{x, C[\mathsf{aMPK}, x']} \leftarrow \mathsf{iKeyGen}(\mathsf{iMSK}, x, C[\mathsf{aMPK}, x'])$. Then, the iFE decryption process only succeeds if both iCT_x and $\mathsf{iSK}_{x, C[\mathsf{aMPK}, x']}$ were generated w.r.t. the same identity x, which is exactly the case where updates are admissible.

Allowing Multiple Update Tokens. The second issue is more difficult to solve, and we only know how to do so under the assumption that the number of updates per ciphertext is bounded, i.e., there is a maximum number Q of tokens $\Delta_{x \to x'}$ that can be issued per attribute x. Intuitively, if the maximum number Q of iFE secret keys that can decrypt a certain iFE ciphertext iCT_x is fixed, then we can encrypt Q independent random strings $r_1, \ldots, r_Q$ in iCT_x, and each secret key can use a different r_i. This naive idea would allow issuing multiple update tokens, but it requires a stateful TokGen algorithm to make sure that each random string r_i is used only once. To circumvent this, we employ a technique based on cover-free sets similarly as [AV19, LLW21]. Specifically, we let the encryption algorithm sample $N = O(Q, \mathsf{poly}(\lambda))$ random strings and encrypt all of them in iCT_x. During the generation of the i-th update token, for $i \in [Q]$, we first pick a random subset $S_i \subseteq [N]$ and then generate an iFE secret key for a slightly modified circuit $C[\mathsf{aMPK}, x']$: instead of running aEnc with fixed random coins r, the new circuit will first compute $r'_i = \sum_{j \in [S]} r_j$ and use r'_i as the randomness of aEnc. Under an appropriate choice of parameters, it can be shown that each of the random subsets $S_1, \ldots, S_Q \subseteq [N]$ contains at least one index $j_i \in S_i$ which does not appear in any of the other sets. Therefore, the strings $r'_1, \ldots, r'_Q$ are all random and independently distributed.

Relying on IPFE. Functional encryption for general circuits is a heavy tool, that even implies indistinguishability obfuscation [BV15], and hence, we would like to avoid it. For this reason, we take a closer look at the circuits $C[\mathsf{aMPK}, x']$, which we generate iFE secret keys for. It turns out that these circuits are essentially the encryption algorithm of the employed ABE scheme, and most ABEs are designed such that encryption is very simple. In particular, nearly all lattice-based ABE schemes have encryption algorithms consisting of two steps.

(1) Sample a set of vectors according to certain distributions (e.g., a uniform vector $\mathbf{s} \in \mathbb{Z}_q^n$ and a Gaussian vector $\mathbf{e} \in \mathbb{Z}^m$).
(2) Compute and output a linear function over the message input μ and the samples chosen in the first step, where the coefficients are determined by the master public key aMPK.

This calls for the use of IPFE as a lightweight replacement for FE for general circuits. We change our previous template as follows. Instead of using a powerful FE scheme that can take care of the entire ABE encryption procedure, we now use an IPFE scheme that performs only step (2). To compensate for this weaker FE

primitive, we must perform step (1) at the time of encryption. That means we directly embed the samples of the required distributions in iCT_x instead of only the random coins that would be used for the sampling at a later stage.

This idea almost works out, but there is one more technical subtlety that we need to take into account. As explained in the previous paragraph, we use the cover-free sets technique to enable the generation of multiple update tokens. In particular, the usage of this technique implies that the samples used during the update process are not the actual samples chosen during encryption, but the *sum* of t such samples for an appropriately chosen $t \in O(\lambda)$. In the case of uniformly random coins $r \in \{0,1\}^{\mathsf{poly}(\lambda)}$ this fact was irrelevant, and it still does not matter for uniform samples $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$. However, it is relevant for Gaussian samples $\mathbf{e} \in \mathbb{Z}^m$ since the sum of several such samples increases the standard deviation. Therefore, we need to choose the parameters of the underlying ABE scheme sufficiently large such that decryption remains correct even for slightly larger error vectors.

Proof Strategy. The security proof is very simple and directly follows from the security of the underlying ABE and IPFE scheme. As we use the IPFE scheme to generate ABE ciphertexts during the update process, it is important to argue that IPFE reveals nothing about the input μ except for the inner products with the vectors of the IPFE secret keys (which have the same distribution as fresh ABE ciphertexts by construction). In other words, we need an IPFE scheme that achieves *SIM-based* security. To our knowledge, there is currently no SIM-secure identity-based IPFE in the literature. We therefore provide such a scheme in Appendix A.1 as an extension of the SIM-secure IPFE presented in [ALMT20].

Single-Hop CU-PE. We now discuss the high-level ideas behind our compiler from CU-ABE to CU-PE. As a starting point, we recall the conversion in the classical ABE setting using a primitive called *lockable obfuscation* (LO) [GKW17, WZ17], which can be instantiated from plain LWE. Roughly speaking, LO allows obfuscating a circuit C with respect to a message μ and a lock value σ. Correctness asks that an evaluation of the obfuscated circuit on some input x yields μ if $C(x) = \sigma$ and $\perp$ otherwise. Simulation security requires that if σ looks random to the adversary, then the obfuscated circuit is computationally indistinguishable from a garbage program that does not carry any information about μ or C.

Recap: From ABE to PE. We first recall the compiler from [GKW17] that lifts a plain ABE scheme into a PE scheme. To encrypt a message μ under an attribute x, we first encrypt a random lock value σ using the ABE encryption algorithm to obtain an ABE ciphertext aCT_x. Then, we construct a circuit $C[\mathsf{aCT}_x]$ that takes as input an ABE secret key aSK and attempts to decrypt the hardwired ciphertext aCT_x. The final PE ciphertext is an obfuscation $\widetilde{C}$ of the circuit $C[\mathsf{aCT}_x]$ with respect to the message μ. For correctness, we observe that if the secret key aSK is authorized, then aCT_x can be decrypted and the lock value σ is recovered correctly inside the obfuscated circuit, so the lock is unlocked and the message μ is revealed. Conversely, if the secret key is not authorized, then the ABE ciphertext aCT_x hides the lock value σ from the view of the adversary. Therefore, we can rely on the security of LO and erase any information inside the obfuscated circuit including the message μ and the attribute x.

Adding Ciphertext Updates. It arises the question as to how ciphertext updates can be integrated into this conversion. The main barrier seems to be that ciphertexts are obfuscated circuits which, by definition, look like a garbage program unless they can be unlocked. We clearly cannot afford to unlock an obfuscated circuit while performing an update as this would mean to lose all security guarantees provided by the obfuscation. Therefore, the only plausible option seems to be that we somehow "encode" the update token and perform the actual update of the underlying CU-ABE only during decryption when the obfuscated circuit is evaluated and unlocked. But finding an appropriate encoding for the update token is tricky due to the following opposing requirements.

– On the one hand, the token must be encoded in such a way that it can perform the delayed update inside the specific obfuscated circuit that we want to update.

– On the other hand, the token itself must remain hidden, as otherwise it could be used to update other ciphertexts as well, including the challenge ciphertext. This corresponds to the security requirement that updated ciphertexts should not leak the token which was used for their update.

The latter requirement precludes the use of classical SKE or PKE schemes as a possible encoding of the update token. Even ABE is not enough due to its all-or-nothing nature that would immediately reveal the entire token in the case of a successful decryption.

Construction. As a solution, we rely once again on the power of FE which allows us to run the ABE update procedure with the update token as a *secret* input. In more detail, let us consider a CU-ABE scheme $\mathsf{uABE} = (\mathsf{uSetup}, \mathsf{uEnc}, \mathsf{uKeyGen}, \mathsf{uTokGen}, \mathsf{uUpdate}, \mathsf{uDec})$ and a FE scheme $\mathsf{iFE} = (\mathsf{iSetup}, \mathsf{iEnc}, \mathsf{iKeyGen}, \mathsf{iDec})$ for general circuits. Our CU-PE scheme is constructed in the same way as the PE scheme in the non-updatable setting except for the following modifications. During encryption, we sample a pair of iFE keys $(\mathsf{iMPK}, \mathsf{iMSK})$ and we publish iMPK along with the obfuscated circuit $\widetilde{C}$. To "update" this ciphertext $(\widetilde{C}, \mathsf{iMPK})$ using a token $\Delta_{x \to x'}$, we actually do not touch $\widetilde{C}$ but only encrypt $\mathsf{iCT} \leftarrow \mathsf{iEnc}(\mathsf{iMPK}, \Delta_{x \to x'})$ and return $(\widetilde{C}, \mathsf{iCT})$ as the updated ciphertext.

Then, the actual crux lies in the definition of the circuit which is obfuscated in $\widetilde{C}$. As before, this circuit hardwires an ABE ciphertext uCT_x and takes an ABE secret key uSK as input. However, uSK is not supposed to decrypt uCT_x itself but an update of it. For this the circuit takes the encrypted update token iCT as second input. Now, to make use of iCT, the circuit needs a corresponding secret key iSK which will be hardwired in the description of the obfuscated circuit. This secret key iSK is generated for a circuit $D[\mathsf{uCT}_x]$ that, given an update token $\Delta_{x \to x'}$ as input, evaluates $\mathsf{uUpdate}$ on the hardwired ciphertext uCT_x and outputs the updated ciphertext $\mathsf{uCT}_{x'}$. Thus, the obfuscated circuit can compute

$$\mathsf{uCT}_{x'} \leftarrow \mathsf{uDec}(\mathsf{iSK}, \mathsf{iCT})$$

to obtain the updated ciphertext $\mathsf{uCT}_{x'}$. Then it proceeds as in the non-updatable case by decrypting $\sigma \leftarrow \mathsf{uDec}(\mathsf{uSK}, \mathsf{uCT}_{x'})$. If decryption succeeds, the correct lock value σ is revealed and the obfuscated circuit unlocks. Otherwise, we can rely on the security of LO to conclude that any information inside the obfuscated circuit is hidden. Importantly, we need to argue that iCT leaks nothing about the update token $\Delta_{x \to x'}$ except for the updated ciphertext $\mathsf{uCT}_{x'}$, even if the obfuscated circuit is unlocked. For this, we need once again FE with SIM-security.

Relying on IPFE. As in the generic construction of CU-ABE, we can analyze the concrete circuits $D[\mathsf{uCT}_x]$ that we actually want to generate iFE secret keys for. $D[\mathsf{uCT}_x]$ takes as input an update token $\Delta_{x \to x'}$ and runs the update procedure of uABE. Inspecting our instantiations of uABE, we can observe that all of them have a very simple update procedure. More precisely, we only need to compute a linear function, so iFE can basically be instantiated with any IPFE. However, we note that the iFE ciphertexts are the encrypted update tokens generated during the update process. If we want to support more than one update per ciphertext, then we need a SIM-secure IPFE in the *multi-challenge* version. It is well-known that SIM-secure IPFE against an unbounded number of challenges is impossible in the standard model [BSW11]. However, we are able to provide a SIM-secure IPFE for a *bounded* number of challenges in Appendix A.3. This eventually gives CU-PE for a bounded number of updates per ciphertext.

Unbounded-Hops CU-ABE from Public-coin Evasive LWE. From the perspective of practical applications, updates are not expected to occur very frequently in most scenarios. For instance, in the case of Cloudflare, the update would be on the policies attached to the ciphertexts, which are normally policies about locations of the servers that are allowed to process the ciphertexts. Therefore, we believe that a bounded number of updates, which has been achieved in our single-hop CU-ABE and CU-PE schemes under plain LWE, does not pose a significant practical limitation. Nevertheless, it remains theoretically significant to explore how to achieve stronger security goals that support unbounded updates, whether in terms of the number of update tokens that can be applied to a specific, non-updated ciphertext, or in terms of the number of hops.

Essential Requirements for Unbounded Updates. Before discussing the construction ideas for supporting unbounded updates, we first review the generic construction of CU-ABE and analyze the reason why it permits only single-hop and a bounded number of update tokens. Intuitively, an updated ciphertext is structured differently from a non-updated ciphertext, preventing it from being further updated by additional update tokens. Specifically, each non-updated ciphertext includes an additional IPFE ciphertext, which provides the encryption randomness and plaintext needed for generating an updated ABE ciphertext. Furthermore, for any given original ciphertext, the randomness of each ciphertext updated from it is derived from the random combination of the randomness embedded in the IPFE ciphertext. However, the cover-free set technique imposes a restriction, such that at most Q resulting updated-ciphertext randomness can remain independent. Consequently, we observe that to achieve multi-hop and allow an unbounded number of update tokens, the following requirements are desirable:

- The updated ciphertext retains the same structure as the original ciphertext before the update.
- The randomness contained in the updated ciphertexts remains independent and unaffected, even after multiple successive updates.

To support an unbounded number of update tokens, it is natural to consider the usage of pseudorandom functions (PRFs) to compute independent ciphertext randomness. However, existing lattice-based PRFs [BPR12, BP14] involve complex computations such as rounding. These computations exceed the power of classical inner-product operations, and hence, require more powerful FE schemes. One might consider the use of noisy linear FE schemes [Agr19, AP20] as a replacement for plain IPFE. This primitive is known to imply FE for NC^1 circuits (under the additional assumption of LWE and PRGs in NC^0), and therefore, is powerful enough to evaluate lattice-based PRFs. Though, the existing constructions [Agr19, AP20] rely on non-standard assumptions and lack a formal security proof. As an alternative, we might consider using FE for pseudorandom functionalities (prFE) recently proposed by Agrawal et al. [AKY24]. Intuitively, our compiler could use prFE to generate ABE ciphertexts which—by the admissibility of the adversary—cannot be decrypted, and thus, are pseudorandom. This solution would indeed allow us to achieve both unbounded hops and tokens. However, the prFE scheme in [AKY24] relies on *private-coin* evasive LWE and a recent work [BÜW24, DJM+25, HJL25, AMYY25] shows a counterexample to this assumption using null-iO. We therefore explore another technical approach relying on *public-coin* evasive LWE, which resists all the counterexamples given in [BÜW24, DJM+25, HJL25, AMYY25]. We will elaborate our construction idea in the following part.

Ideas from Updatable Encryption Schemes. We now begin by recalling updatable encryption schemes [Jia20, Nis22] that support multi-hop updates. To update a ciphertext from pk_1 to pk_2, an update token $\Delta_{1\to 2}$ is typically generated using both sk_1 and pk_2. During the updating procedure, the one-time pad (OTP) in the original ciphertext ct_1 (encrypted under pk_1) is removed by the secret key sk_1, which is embedded within the update token $\Delta_{1\to 2}$. Simultaneously, a fresh OTP is attached to the resulting updated ciphertext uct_2. Given the update token $\Delta_{1\to 2}$ and ciphertext ct_1, the update algorithm can successfully compute the updated ciphertext, provided that the input token and ciphertext are valid matches.

To perform a similar transformation in the case of CU-ABE, taking KP-ABE as an example, an attribute-related secret key is required to play the role of sk_1. Consider the classical LWE-based KP-ABE scheme of [BGG+14], a ciphertext $\mathsf{ct}_{\mathbf{x}}$ contains the following three components (where we use underlines to denote noise terms):

$$\beta_0 = \underline{\mathbf{A}^\top \mathbf{s}},\ \beta_1 = \underline{\mathbf{u}^\top \mathbf{s}} + \lceil q/2 \rceil \mu,\ \ \mathbf{c}_{\mathsf{attr}} = \underline{(\mathbf{A}_{\mathsf{attr}} + \mathbf{x}^\top \otimes \mathbf{G})^\top \mathbf{s}},$$

for a uniformly random matrix $\mathbf{A}$, a uniformly random vector $\mathbf{u}$, an attribute encoding matrix $\mathbf{A}_{\mathsf{attr}}$ and a LWE secret $\mathbf{s}$. A short preimage $\mathbf{r}_{\mathbf{x}}$ satisfying $[\mathbf{A} \mid \mathbf{A}_{\mathsf{attr}} + \mathbf{x} \otimes \mathbf{G})] \cdot \mathbf{r}_{\mathbf{x}} = \mathbf{u}$ can be served as the secret-key for $\mathbf{x}$. However, an ABE ciphertext $\mathsf{ct}_{\mathbf{x}}$ can be fully decrypted using $\mathbf{r}_{\mathbf{x}}$, the security would thus be broken if the adversary could obtain $\mathbf{r}_{\mathbf{x}}$ itself. Therefore, we turn to construct the update token $\Delta_{\mathbf{x}\to\mathbf{x}'}$ as an encryption of $\mathbf{r}_{\mathbf{x}}$ for the attribute $\mathbf{x}'$:

$$\mathbf{D}_0 = \underline{\mathbf{A}^\top \mathbf{S}},\ \mathbf{D}_1 = \underline{\mathbf{u}^\top \mathbf{S}} - \mathbf{r}_{\mathbf{x}}^\top \mathbf{G},\ \ \mathbf{D}_{\mathsf{attr}} = \underline{(\mathbf{A}_{\mathsf{attr}} + \mathbf{x}'^\top \otimes \mathbf{G})^\top \mathbf{S}}$$

Given an update token $\Delta_{\mathbf{x}\to\mathbf{x}'}$ and a ciphertext $\mathsf{ct}_{\mathbf{x}}$, the updated ciphertext $\mathsf{uct}_{\mathbf{x}'}$ can be computed by first multiplying components $(\mathbf{D}_0, \mathbf{D}_{\mathsf{attr}})$ in $\Delta_{\mathbf{x}\to\mathbf{x}'}$, respectively with the bit-decomposition $\mathbf{G}^{-1}$ of $(\beta_0, \mathbf{c}_{\mathsf{attr}})$, resulting in LWE instances $(\beta_0', \mathbf{c}'_{\mathsf{attr}})$ sharing the same secret $\mathbf{s}' = \mathbf{S}\mathbf{G}^{-1}(\beta_0, \mathbf{c}_{\mathsf{attr}})$. Subsequently, the payload component of the updated ciphertext is computed by summing β_1 from $\mathsf{ct}_{\mathbf{x}}$ with $\mathbf{D}_1 \cdot \mathbf{G}^{-1}(\beta_0, \mathbf{c}_{\mathsf{attr}})$. More precisely, the multiplication of $\underline{\mathbf{u}^\top\mathbf{S}} - \mathbf{r}_{\mathbf{x}}^\top\mathbf{G}$ and $\mathbf{G}^{-1}(\beta_0, \mathbf{c}_{\mathsf{attr}})$ results in $\underline{\mathbf{u}^\top\mathbf{s}'} - \mathbf{u}^\top\mathbf{s}$. By adding β_1, the original message encoding is preserved, while the original OTP in β_1 is canceled, resulting in $\beta_1' = \underline{\mathbf{u}^\top\mathbf{s}'} + \lceil q/2 \rfloor \mu$. Additionally, the updated ciphertext $\mathsf{uct}_{\mathbf{x}'} = (\beta_0', \beta_1', \mathbf{c}'_{\mathsf{attr}})$ maintains the same structure as a non-updated ciphertext, enabling multi-hop updates. Moreover, the resulting ciphertext randomness, derived from both the token and the non-updated ciphertext $\mathsf{ct}_{\mathbf{x}}$, is independent from the number of tokens applied on $\mathsf{ct}_{\mathbf{x}}$. However, the error terms in the updated ciphertext increase with each update, requiring us to adjust the parameter of modulus to ensure decryption correctness even for multi-hop updates.

Issues in Security Proof. Unlike traditional ABE schemes that do not consider ciphertext updatability, we need to additionally handle update token queries, which can be viewed as encryptions of short preimages.

During the security proof, the attribute-encoding matrix $\mathbf{A}_{\mathsf{attr}}$ is typically set as $\mathbf{A}\mathbf{R}^* - \mathbf{x}^{*\top}\mathbf{G}$, by inserting in the challenge attribute $\mathbf{x}^*$ and sampling a random matrix $\mathbf{R}^*$ with small norm. This programmed result then enables us to simulate any admissible secret key queries and generate short preimages $\mathbf{r}_{\mathbf{x}}$ for $\mathbf{x} \neq \mathbf{x}^*$ by relying on the public trapdoor of the gadget matrix $\mathbf{G}$, along with the secret matrix $\mathbf{R}^*$. Additionally, $\mathbf{r}_{\mathbf{x}^*}$ can also be simulated without access to either the trapdoor of $\mathbf{A}$ or $\mathbf{G}$. In more detail, we can first sample $\mathbf{r}_{\mathbf{x}^*}$ from particular Gaussian distribution, and then set the image vector $\mathbf{u}$ to satisfy $\mathbf{u} := [\mathbf{A} \mid \mathbf{A}_{\mathsf{attr}} + \mathbf{x}^{*\top}\mathbf{G}] \cdot \mathbf{r}_{\mathbf{x}^*} = [\mathbf{A} \mid \mathbf{A}\mathbf{R}^*] \cdot \mathbf{r}_{\mathbf{x}^*} = \mathbf{A}[\mathbf{I} \mid \mathbf{R}^*] \cdot \mathbf{r}_{\mathbf{x}^*}$. In order to rely on LWE, the payload of the challenge ciphertext can be set as $\beta_1^\top = (\underline{\mathbf{s}^\top\mathbf{A}})[\mathbf{I} \mid \mathbf{R}^*] \cdot \mathbf{r}_{\mathbf{x}^*} + \lceil q/2 \rceil \cdot \mu$.

The problem arises when simulating the token queries that update to attributes $\mathbf{x}' \neq \mathbf{x}^*$. More precisely, when reducing the security to the LWE assumption, we cannot simulate the update token $\Delta_{\cdot\to\mathbf{x}'}$, especially the attribute-encoding component $\mathbf{D}_{\mathsf{attr},\mathbf{x}'} = \underline{(\mathbf{A}\mathbf{R}^* + (\mathbf{x}' - \mathbf{x}^*)^\top \otimes \mathbf{G})^\top\mathbf{S}}$, on given the LWE sample $(\mathbf{A}, \underline{\mathbf{A}^\top\mathbf{S}})$. Obviously, the lack of the corresponding LWE secret $\mathbf{S}$ makes it impossible to correctly compute $((\mathbf{x}' - \mathbf{x}^*)^\top \otimes \mathbf{G})^\top\mathbf{S}$. In other words, update tokens $\Delta_{\cdot\to\mathbf{x}'}$ cannot be proven as pseudorandom. Therefore, the short preimage inside the update tokens $\Delta_{\cdot\to\mathbf{x}'}$ also cannot be proven to be hidden. However, we will not be able to to prove the ciphertext component $\beta_1^\top$ constructed as above as pseudorandom if $[\mathbf{I} \mid \mathbf{R}^*] \cdot \mathbf{r}_{\mathbf{x}^*}$ is not known to the adversary, as otherwise Leftover Hash Lemma (LHL) would not apply. Unfortunately, as noted earlier, we cannot show that $\mathbf{r}_{\mathbf{x}^*}$ is hidden by claiming the pseudorandomness of update tokens $\Delta_{\mathbf{x}^*\to\mathbf{x}'}$ containing $\mathbf{r}_{\mathbf{x}^*}$.

Relying on Public-Coin Evasive LWE. As analyzed above, the key to prove security is to enable to generate short preimages (related to the adversary's different queries), and in the meanwhile, to ensure the necessary pseudorandomness (to prevent leakage). Fortunately, the *public-coin* evasive LWE assumption [Wee22, Tsa22] is exactly such an assumption, that claims the pseudorandomness of the view of the adversary considering the existence of preimage. More precisely, the public-coin evasive LWE assumption asserts that if the joint distribution of $(\mathbf{B}, \mathbf{P}, \mathbf{B}^\top\mathbf{s} + \mathbf{e}, \mathbf{P}^\top\mathbf{s} + \mathbf{e}')$ is computationally close to uniformly random, then the LWE sample $\mathbf{B}^\top\mathbf{s} + \mathbf{e}$ remains pseudorandom even if the adversary is given $\mathbf{B}^{-1}(\mathbf{P})$, which maps $\mathbf{B}$ to $\mathbf{P}$. Intuitively, by relying on the public-coin evasive LWE assumption, we can prove the security by showing that, during the security game, the adversary's transformed view of the transcript is pseudorandom. The public-coin evasive LWE assumption naturally enables us to handle all preimage sampling without removing the trapdoor, simplifying the proof process. Furthermore, the pseudorandomness of update tokens which formed as encryptions of preimages $\mathbf{r}_{\mathbf{x}}$, can also be guaranteed under the public-coin evasive LWE.

Concretely, during the security proof, we first assert that the distribution resulting solely from the queried secret keys and the challenge ciphertext is close to uniform. We then extend this indistinguishability result to all update tokens that may appear in the update chain starting from the challenge attribute, given the similar structure of the ciphertexts. Finally, the honestly updated ciphertexts can be switched to uniform, as the inputs for generating these updated ciphertexts have already been replaced by random.

Consequently, we obtain both a CU-KP-ABE scheme and a CU-CP-ABE scheme by applying the aforementioned updating approach on ABE schemes from [HLL23b, Wee22], and by relying on the public-coin evasive LWE assumption.

2 Preliminaries

2.1 Notational Conventions

Let $\lambda \in \mathbb{N}$ be the security parameter. Except in the definitions, we will suppress λ in subscripts for brevity. A nonnegative function $\varepsilon\colon \mathbb{N} \to \mathbb{R}$ is negligible if $\varepsilon(\lambda) = O(\lambda^{-n})$ for all $n \in \mathbb{N}$. An algorithm A is said to be *efficient* if it runs in probabilistic polynomial time (PPT) in the security parameter. We write $x \leftarrow \mathsf{A}(y)$ to indicate that x is the random variable distributed according to the output of A on input y and random coins r. When we want to make the randomness r used by A explicit, we write it as $x \leftarrow \mathsf{A}(y; r)$. If A is a deterministic polynomial time (DPT) algorithm, we use the notation $x := \mathsf{A}(y)$. If A has oracle access to a procedure O, we represent this by superscript A^{O}.

To avoid confusion, we always write vectors $\mathbf{v}$ and matrices $\mathbf{A}$ in boldface and use uppercase letters for the latter. Scalars s are written in italics. Unless otherwise stated, all vectors $\mathbf{v}$ are viewed as *column* vectors. The corresponding row vector is denoted by $\mathbf{v}^\top$.

The Euclidean norm (known as ℓ_2 norm) and infinity norm of a vector is written as $\|\mathbf{v}\|$ and $\|\mathbf{v}\|_\infty$, respectively. Additionally, we use $\widetilde{\mathbf{A}}$ to denote its Gram-Schmidt orthogonalization. The Euclidean norm and spectral norm of a matrix $\mathbf{A}$ is denoted by $\|\mathbf{A}\|$ and $s_1(\mathbf{A})$, respectively. Some more notation on security experiments and distributions can be found in Section 2.1.

Sets and Indexing. We denote by $\mathbb{Z}$ and $\mathbb{N}$ the sets of integers and natural numbers (positive integers). For integers m and n, we write $[m; n]$ to denote the set $\{z \in \mathbb{Z} : m \le z \le n\}$ and let $[n] := [1; n]$. For a prime number p, $\mathbb{Z}_p$ denotes the finite field of integers modulo p. To index a vector or the columns of a matrix, we write $\mathbf{v}[i]$ and $\mathbf{A}[j]$. In contrast, objects of some collection that is not regarded as a vector or matrix are indexed using subscripts (or superscripts in some cases). For instance, $\mathbf{v}_i$ represents a vector, not a component of some vector. If i runs through some index set $[n]$, it means that there are n vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$. If the n objects are scalars (or not explicitly vectors), we will write $v_1, \ldots, v_n$ instead.

Security Experiments and Distributions. Let Exp be an interactive *experiment* that interacts with an algorithm $\mathcal{A}$ (called the *adversary*), depends on the security parameter λ and has binary outcome. We also refer to such objects as *games* or *hybrids*. We let "$\mathsf{Exp}_{\mathcal{A}}(1^\lambda) \to 1$" denote the event that the outcome of running Exp with $\mathcal{A}$ on security parameter λ is 1. For two experiments Exp^0 and Exp^1, we define the distinguishing advantage of $\mathcal{A}$ against the tuple $(\mathsf{Exp}^0, \mathsf{Exp}^1)$ as

$$\mathsf{Adv}_{\mathsf{Exp}^0,\mathsf{Exp}^1,\mathcal{A}}(\lambda) := \left|\Pr\left[\mathsf{Exp}^1_{\mathcal{A}}(1^\lambda) \to 1\right] - \Pr\left[\mathsf{Exp}^0_{\mathcal{A}}(1^\lambda) \to 1\right]\right| .$$

We write $\mathsf{Exp}^0 \approx_c \mathsf{Exp}^1$ if the experiments are *computationally indistinguishable*, i.e., their distinguishing advantage is negligible for all efficient adversaries $\mathcal{A}$. We write $\mathsf{Exp}^0 \approx_s \mathsf{Exp}^1$ if the experiments are *statistically indistinguishable*, i.e., their distinguishing advantage is negligible for all (even unbounded) adversaries. We write $\mathsf{Exp}^0 \equiv \mathsf{Exp}^1$ if the experiments are *identically distributed*, i.e., their distinguishing advantage is 0 for all (even unbounded) adversaries. By default, the term *indistinguishable* refers to computational indistinguishability.

More general, the same notations can be used for sequences of distributions. Let $D^0 = \{D^0_\lambda\}_{\lambda\in\mathbb{N}}$ and $D^1 = \{D^1_\lambda\}_{\lambda\in\mathbb{N}}$ be two sequences of distributions. For $b \in \{0, 1\}$, we define $\mathsf{Exp}^b_{\mathcal{A}}(1^\lambda)$ as follows: sample $x \xleftarrow{\$} D^b_\lambda$, run $\mathcal{A}(1^\lambda, x)$ and use the output of $\mathcal{A}$ as the outcome of the experiment. Then we write $D^0 \approx_c D^1$ (resp. $D^0 \approx_s D^1$, $D^0 \equiv D^1$) if $\mathsf{Exp}^0_{\mathcal{A}} \approx_c \mathsf{Exp}^1_{\mathcal{A}}$ (resp. $\mathsf{Exp}^0_{\mathcal{A}} \approx_s \mathsf{Exp}^1_{\mathcal{A}}$, $\mathsf{Exp}^0_{\mathcal{A}} \equiv \mathsf{Exp}^1_{\mathcal{A}}$).

2.2 Tensor Product

For matrices $\mathbf{A} = (a_{i,j}) \in \mathbb{Z}^{n_1 \times m_1}$ and $\mathbf{B} \in \mathbb{Z}^{n_2 \times m_2}$, the tensor product is defined as

$$\mathbf{A} \otimes \mathbf{B} = \begin{pmatrix} a_{1,1}\mathbf{B} & \cdots & a_{1,m_1}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{n_1,1}\mathbf{B} & \cdots & a_{n_1,m_1}\mathbf{B} \end{pmatrix} \in \mathbb{Z}^{n_1 n_2 \times m_1 m_2}.$$

We will use the mixed-product property for tensor product that

$$(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = (\mathbf{AC}) \otimes (\mathbf{BD}).$$

In particular, for any pair of column vectors $\mathbf{a}, \mathbf{b} \in \mathbb{Z}^n$, we have

$$\mathbf{a} \otimes \mathbf{b} = (\mathbf{a} \otimes \mathbf{I}_n)(1 \otimes \mathbf{b}) = (\mathbf{a} \otimes \mathbf{I}_n)\mathbf{b}.$$

2.3 Lattices

Let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ be arbitrary matrix for some positive integers n, m, q, define the full-rank m-dimensional q-ary lattices as follows:

$$\Lambda(\mathbf{A}) = \left\{ \mathbf{z} \in \mathbb{Z}^m \mid \exists\ \mathbf{s} \in \mathbb{Z}_q^n,\ s.t.\ \mathbf{A}^\top \mathbf{s} = \mathbf{z} \mod q \right\}$$
$$\Lambda^\perp(\mathbf{A}) = \{ \mathbf{z} \in \mathbb{Z}^m \mid \mathbf{Az} = \mathbf{0} \mod q \}.$$

For a fixed $\mathbf{u} \in \mathbb{Z}_q^n$, define a coset of $\Lambda^\perp$ as:

$$\Lambda_{\mathbf{u}}^\perp(\mathbf{A}) = \{ \mathbf{z} \in \mathbb{Z}^m \mid \mathbf{Az} = \mathbf{u} \mod q \}.$$

The Learning with Errors problem, or LWE, is the problem of distinguishing noisy inner products from random.

Definition 1 (LWE **[Reg05]).** *Let $n \geq 1$ and $q \geq 2$ be integers, and let $\mathcal{X}$ be a probability distribution on $\mathbb{Z}_q$. For $\mathbf{s} \in \mathbb{Z}_q$, let $A_{\mathbf{s},\mathcal{X}}$ be the probability distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by choosing a vector $\mathbf{a} \in \mathbb{Z}_q$ uniformly at random, choosing $e \in \mathbb{Z}_q$ according to $\mathcal{X}$, and outputting $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e)$.*

The decision-$\mathsf{LWE}_{q,n,\mathcal{X}}$ *problem is: for uniformly random $\mathbf{s} \in \mathbb{Z}_q$, given a $\mathsf{poly}(n)$ number of samples that are either (all) from $A_{\mathbf{s},\mathcal{X}}$ or (all) uniformly random in $\mathbb{Z}_q^n \times \mathbb{Z}_q$, output 0 if the former holds and 1 if the latter holds.*

Proposition 1 (LWE **hardness [Reg05, GPV08]).** *Let $\alpha = \alpha(n) \in (0,1)$ and let $q = q(n)$ be a prime such that $\alpha \cdot q > 2\sqrt{n}$. If there exists an efficient (possibly quantum) algorithm that solves $\mathsf{LWE}_{q,n,\mathcal{D}_{\mathbb{Z},\alpha q}}$, then there exists an efficient quantum algorithm for approximating SIVP and GapSVP in the ℓ_2 norm, in the worst case, to within $\widetilde{O}(n/\alpha)$ factors.*

Gaussian on Lattices. For any $s > 0$, the Gaussian distribution $\mathcal{D}_{s,\mathbf{c}}$ centered at $\mathbf{c} \in \mathbb{R}^n$ with parameter s is defined by the following probability distribution function:

$$\forall \mathbf{x} \in \mathbb{Z}^n,\ \rho_{s,\mathbf{c}}(\mathbf{x}) = \exp(-\pi \left\| \mathbf{x} - \mathbf{c} \right\|^2 / s^2).$$

Usually, we omit the subscripts $\mathbf{c}$ when it is taken to be $\mathbf{0}$.

Let $s > 0$, $\mathbf{c} \in \mathbb{R}^n$, for any n-dimensional lattice Λ, the *discrete Gaussian distribution over* Λ is defined as

$$\forall \mathbf{x} \in \Lambda, \mathcal{D}_{\Lambda,s,\mathbf{c}} = \frac{\rho_{s,\mathbf{c}}(\mathbf{x})}{\rho_{s,\mathbf{c}}(\Lambda)}.$$

Gaussian Samplings. The gadget matrix $\mathbf{G} \in \mathbb{Z}_q^{n\times m}$ is a primitive matrix defined by gadget vector $\mathbf{g}$ as $\mathbf{G} := \mathbf{I}_n \otimes \mathbf{g}^\top \in \mathbb{Z}_q^{n\times nk}$. We usually consider gadget vector $\mathbf{g}^\top := [1\ 2\ 4\ \cdots\ 2^{k-1}] \in \mathbb{Z}_q^{1\times k}$, where $k = \lceil \log_2 q \rceil$. Given a vector $\mathbf{c} \in \mathbb{Z}_q^n$, we denote the vector $\mathbf{c}' = (\mathbf{c}'_1, \ldots, \mathbf{c}'_n) \in \{0,1\}^{nk}$, where each $\mathbf{c}'_i$ is the bit representation of $\mathbf{c}[i]$. It holds that $\mathbf{G}\mathbf{G}^{-1}(\mathbf{c}) = \mathbf{c}$.

Lemma 1 (TrapGen [MP12]). *Let $q, n, m \in \mathbb{Z}$ with $q \geq 2$ and $m \geq 6n\log q$. There is a probabilistic polynomial-time algorithm $\mathsf{TrapGen}(1^n, 1^m, q)$ that with overwhelming probability (in n) outputs a pair $(\mathbf{A} \in \mathbb{Z}_q^{n\times m}, \mathbf{T} \in \mathbb{Z}^{m\times m})$ such that $\mathbf{A}$ is statistically close to uniform in $\mathbb{Z}_q^{n\times m}$ and $\mathbf{T}$ is a basis for $\Lambda^\perp(\mathbf{A})$ satisfying $\left\|\widetilde{\mathbf{T}}\right\| \leq O(\sqrt{n\log q})$ and $\|\mathbf{T}\| \leq O(n\log q)$.*

Lemma 2 (SamplePre [GPV08, MP12]). *There is an efficient algorithm $\mathsf{SamplePre}$ that with overwhelming probability over all random choices, does the following: For any $\mathbf{u} \in \mathbb{Z}_q^n$, and large enough $s = O(\sqrt{n\log q})$, the randomized algorithm $\mathsf{SamplePre}(\mathbf{A}, \mathbf{T}_\mathbf{A}, \mathbf{u}, s)$ outputs a vector $\mathbf{r} \in \mathbb{Z}^m$ with norm $\|\mathbf{r}\|_\infty \leq \|\mathbf{r}\| \leq s\sqrt{n}$ with overwhelming probability such that $\mathbf{A}\cdot\mathbf{r} = \mathbf{u}$.*

Lemma 3 (SampleLeft [ABB10]). *Let $q > 2$, $\mathbf{A}, \mathbf{B} \in \mathbb{Z}_q^{n\times m}$ be two full rank matrices with $m > n$, $\mathbf{T}_\mathbf{A}$ be a trapdoor matrix for $\mathbf{A}$, a matrix $\mathbf{U} \in \mathbb{Z}_q^{n\times l}$ and $s \geq \left\|\widetilde{\mathbf{T}_\mathbf{A}}\right\| \cdot \omega(\sqrt{\log m})$. Then there exists a PPT algorithm $\mathsf{SampleLeft}(\mathbf{A}, \mathbf{B}, \mathbf{T}_\mathbf{A}, \mathbf{U}, s)$ that outputs a matrix $\mathbf{K} \in \mathbb{Z}_q^{2m\times l}$, which is distributed statistically close to $\mathcal{D}_{\Lambda^\perp_\mathbf{U}(\mathbf{A}|\mathbf{B}),s}$.*

Lemma 4 (SampleRight [MP12]). *Let $q > 2$, $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ be a full rank matrices with $m > n$, $\mathbf{R} \in \mathbb{Z}^{m\times m}$, $\mathbf{U} \in \mathbb{Z}_q^{n\times l}$, $\gamma \in \mathbb{Z}_q$ with $\gamma \neq 0$ and $s \geq \sqrt{5}\cdot s_1(\mathbf{R}) \cdot \omega\sqrt{\log m}$. Then there exists a PPT algorithm $\mathsf{SampleRight}(\mathbf{A}, \mathbf{R}, \gamma, \mathbf{U}, s)$ that outputs a matrix $\mathbf{K} \in \mathbb{Z}_q^{2m\times l}$, which is distributed statistically close to $\mathcal{D}_{\Lambda^\mathbf{U}_q(\mathbf{A}|\mathbf{A}\cdot\mathbf{R}+\gamma\mathbf{G}),s}$.*

Lemma 5 ([DM14]). *Let $\mathbf{X} \in \mathbb{R}^{n\times m}$ be a subgaussian random matrix with parameter s. There exists a universal constant $c \approx 1/\sqrt{2\pi}$ such that for any $t > 0$, we have $s_1(\mathbf{X}) \leq c\cdot s\cdot(\sqrt{m} + \sqrt{n} + t)$ except with probability at most $2/e^{\pi t^2}$.*

Lemma 6 (Leftover Hash Lemma [ABB10]). *Suppose that $m > (n+1)\log q + \omega(\log n)$ and that $q > 2$ is prime. Let $\mathbf{R}$ be an $m\times k$ matrix chosen uniformly in $\{-1,1\}^{m\times k} \mod q$, where $k = k(n)$ is polynomial in n. Let $\mathbf{A}$ and $\mathbf{B}$ be matrices chosen uniformly in $\mathbb{Z}_q^{n\times m}$ and $\mathbb{Z}_q^{n\times k}$ respectively. Then, for all vectors $\mathbf{e} \in \mathbb{Z}_q^m$, the distribution $(\mathbf{A}, \mathbf{A}\mathbf{R}, \mathbf{R}^\top\mathbf{e})$ is statistically close to the distribution $(\mathbf{A}, \mathbf{B}, \mathbf{R}^\top\mathbf{e})$.*

Lemma 7 ([GPV08]). *For any prime q, integers $n \geq 1$, $m \geq 2n\log q$, $s \geq \omega(\sqrt{m})$, the following two distributions are statistically indistinguishable:*

- $(\mathbf{A}, \mathbf{u}, \mathbf{y})$: $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^n$, $\mathbf{y} \leftarrow \mathcal{D}_{\Lambda^\mathbf{u}_q,s}$.
- $(\mathbf{A}, \mathbf{u}, \mathbf{y})$: $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, $\mathbf{y} \leftarrow \mathcal{D}_{\mathbb{Z}^m,s}$, $\mathbf{u} = \mathbf{A}\mathbf{y} \mod q$.

Lemma 8 ([MR04, GPV08]). *For any n-dimensional lattice Λ, let $\mathbf{B}$ be a basis for Λ. Suppose $s \geq \left\|\widetilde{\mathbf{B}}\right\| \cdot \omega(\sqrt{\log n})$, then $\Pr[\|\mathbf{x}\| \geq s\sqrt{n} : \mathbf{x} \leftarrow \mathcal{D}_{\Lambda,s}] \leq \mathsf{negl}(n)$.*

Lemma 9 (Noise flooding [GKPV10], Lemma 3). *Let $n \in \mathbb{N}$. For any real $\sigma = \omega(\sqrt{\log n})$, and any $\mathbf{c} \in \mathbb{Z}^n$,*

$$\mathsf{SD}(\mathcal{D}_{\mathbb{Z}^n,\sigma}, \mathcal{D}_{\mathbb{Z}^n,\sigma,\mathbf{c}}) \leq \|\mathbf{c}\|/\sigma$$

Lemma 10 (Noise Rerandomization [KY16]). *Let q, ℓ, m be positive integers and r a positive real satisfying $r > \max\{\eta_\epsilon(\mathbb{Z}^m), \eta_\epsilon(\mathbb{Z}^\ell)\}$. Let $\mathbf{b} \in \mathbb{Z}_q^m$ be arbitrary and $\mathbf{x}$ chosen from $\mathcal{D}_{\mathbb{Z}^m,r}$. Then for any $\mathbf{V} \in \mathbb{Z}^{m\times\ell}$ and positive real $\sigma > s_1(\mathbf{V})$, there exists a PPT algorithm $\mathsf{ReRand}(\mathbf{V}, \mathbf{b}+\mathbf{x}, r, \sigma)$ that outputs $\mathbf{b}' = \mathbf{b}\mathbf{V} + \mathbf{x}'$ where the statistical distance of the discrete Gaussian $\mathcal{D}_{\mathbb{Z}^\ell,2r\sigma}$ and the distribution of $\mathbf{x}'$ is within 8ϵ.*

Definition 2 ([ABB10] Identity Encoding Matrices). *Let q be a prime and $n \in \mathbb{N}$. A function $\mathsf{H} : \mathbb{Z}_q^n \to \mathbb{Z}_q^{n\times n}$ is said to be an encoding with full-rank differences when H is computable in polynomial time in $n \log q$ and the matrix $\mathsf{H}(u) - \mathsf{H}(v) \in \mathbb{Z}_q^{n\times n}$ has full-rank for all distinct $n, v \in \mathbb{Z}_q^n$.*

Lemma 11 (Truncated discrete Gaussians). *Let $B \geq 0$, the distribution $\mathcal{D}_{\mathbb{Z},\sigma,\leq B}$ is sampled by first sampling $x \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\sigma}$. If $|x| \leq B$, then x is returned, else 0. Let $\sigma \geq 1$ and $B = \sigma\Theta(\sqrt{\lambda})$, then $\mathcal{D}_{Z,\sigma,\leq B}$ is $2^{-\Omega(\lambda)}$-close to $\mathcal{D}_{Z,\sigma}$.*

Theorem 1 ([BGG+14]). *There exist efficient deterministic algorithms EvalF and EvalFX such that for all $n, q, \ell \in \mathbb{N}$, and for any sequence of matrices $(\mathbf{A}_1, \ldots, \mathbf{A}_\ell) \in (\mathbb{Z}_q^{n\times n\lceil \log q\rceil})^\ell$, for any depth-$d$ Boolean circuit $f : \{0,1\}^\ell \to \{0,1\}$ and for every $\mathbf{x} = (x_1, \ldots, x_\ell) \in \{0,1\}^\ell$, the following properties hold.*

- $\mathbf{H}_f \coloneqq \mathsf{EvalF}(f, \mathbf{A}_1, \ldots, \mathbf{A}_\ell)$, $\mathbf{H}_{f,\mathbf{x}} \coloneqq \mathsf{EvalFX}(f, \mathbf{x}, \mathbf{A}_1, \ldots, \mathbf{A}_\ell)$
- $\mathbf{H}_f$, $\mathbf{H}_{f,\mathbf{x}} \in \mathbb{Z}^{(\ell n\lceil \log q\rceil)\times n\lceil \log q\rceil}$
- $\|\mathbf{H}_f\|_\infty, \|\mathbf{H}_{f,\mathbf{x}}\| \leq (n \log q)^{O(d)}$
- $[\mathbf{A}_1 + x_1\mathbf{G}| \ldots |\mathbf{A}_\ell + x_\ell\mathbf{G}] \cdot \mathbf{H}_{f,\mathbf{x}} = [\mathbf{A}_1| \ldots |\mathbf{A}_\ell] \cdot \mathbf{H}_f + f(\mathbf{x})\mathbf{G} \mod q$

Definition 3 (Public-Coin Evasive LWE [BÜW24]). *Let the parameters*

$$\mathsf{param} := (q, n, m, m_A, m_P, t, \mathcal{D}, \mathcal{S}, \chi_A, \chi_B, \chi_P)$$

be parameterized by λ, where $\mathcal{D} \sim \mathbb{Z}_q^{n\times m}, \mathcal{S} \sim \mathbb{Z}_q^{t\times n}, \chi_A \sim \mathbb{Z}^{t\times m_A}, \chi_B \sim \mathbb{Z}^{t\times m}, \chi_P \sim \mathbb{Z}^{t\times m_P}$ are (uniform) distributions. Let Samp be a PPT sampler algorithm that on input 1^λ, outputs

$$(\mathbf{A} \in \mathbb{Z}_q^{n\times m_A}, \mathbf{P} \in \mathbb{Z}_q^{n\times m_P}, \mathsf{aux} \in \{0,1\}^*).$$

Let $\mathsf{Pre}_{\mathcal{A}}^b$ and $\mathsf{Post}_{\mathcal{B}}^b$ be the experiments defined as follows

$\mathsf{Pre}_{\mathcal{A}}^b(1^\lambda)$	$\mathsf{Post}_{\mathcal{B}}^b(1^\lambda)$
$\mathbf{B} \leftarrow\!\$\ \mathcal{D}$	$\mathbf{B} \leftarrow\!\$\ \mathcal{D}$
$(\mathbf{A}, \mathbf{P}, \mathsf{aux}) \leftarrow \mathsf{Samp}(1^\lambda; \mathsf{rand})$	$(\mathbf{A}, \mathbf{P}, \mathsf{aux}) \leftarrow \mathsf{Samp}(1^\lambda; \mathsf{rand})$
assert $\mathbf{P} \in \mathbf{B}\mathbb{Z}^{m\times m_P}$	**assert** $\mathbf{P} \in \mathbf{B}\mathbb{Z}^{m\times m_P}$
$\mathbf{S} \leftarrow\!\$\ \mathcal{S}$	$\mathbf{S} \leftarrow\!\$\ \mathcal{S}$
if $b = 0$ **then**	**if** $b = 0$ **then**
$\quad \mathbf{E}_A \leftarrow\!\$\ \chi_A, \mathbf{E}_B \leftarrow\!\$\ \chi_B, \mathbf{E}_P \leftarrow\!\$\ \chi_P$	$\quad \mathbf{E}_A \leftarrow\!\$\ \chi_A, \mathbf{E}_B \leftarrow\!\$\ \chi_B$
$\quad \mathbf{C}_A \coloneqq \mathbf{SA} + \mathbf{E}_A \mod q$	$\quad \mathbf{C}_A \coloneqq \mathbf{SA} + \mathbf{E}_A \mod q$
$\quad \mathbf{C}_B \coloneqq \mathbf{SB} + \mathbf{E}_B \mod q$	$\quad \mathbf{C}_B \coloneqq \mathbf{SB} + \mathbf{E}_B \mod q$
$\quad \mathbf{C}_P \coloneqq \mathbf{SP} + \mathbf{E}_P \mod q$	$\quad \mathbf{U} \leftarrow\!\$\ \mathbf{B}^{-1}(\mathbf{P})$
if $b = 1$ **then**	**if** $b = 1$ **then**
$\quad \mathbf{C}_A \leftarrow\!\$\ \mathbb{Z}_q^{t\times m_A}$	$\quad \mathbf{C}_A \leftarrow\!\$\ \mathbb{Z}_q^{t\times m_A}$
$\quad \mathbf{C}_B \leftarrow\!\$\ \mathbb{Z}_q^{t\times m}$	$\quad \mathbf{C}_B \leftarrow\!\$\ \mathbb{Z}_q^{t\times m}$
$\quad \mathbf{C}_P \leftarrow\!\$\ \mathbb{Z}_q^{t\times m_P}$	$\quad \mathbf{U} \leftarrow\!\$\ \mathbf{B}^{-1}(\mathbf{P})$
return $\mathcal{A}(\mathsf{rand}, \mathbf{B}, \mathbf{C}_A, \mathbf{C}_B, \mathbf{C}_P)$	**return** $\mathcal{B}(\mathsf{rand}, \mathbf{B}, \mathbf{C}_A, \mathbf{C}_B, \mathbf{U})$

where $\mathbf{B}\mathbb{Z}$ denotes the $\mathbb{Z}$-span of $\mathbf{B}$ and $\mathbf{U} \leftarrow\!\$\ \mathbf{B}^{-1}(\mathbf{P})$ denotes sampling a uniform short pre-image with standard deviation $O(\sqrt{m \log q})$ satisfying $\mathbf{BU} = \mathbf{P}$. Let

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Pre}}(\lambda) := \left|\Pr\left[\mathsf{Pre}_{\mathcal{A}}^0(1^\lambda) = 1\right] - \Pr\left[\mathsf{Pre}_{\mathcal{A}}^1(1^\lambda) = 1\right]\right|,$$

$$\mathsf{Adv}^{\mathsf{Post}}_{\mathcal{B}}(\lambda) := \left|\Pr\left[\mathsf{Post}^0_{\mathcal{B}}(1^\lambda)=1\right] - \Pr\left[\mathsf{Post}^1_{\mathcal{B}}(1^\lambda)=1\right]\right|.$$

The $\mathsf{PublicEvLWE}_{\mathsf{param}}$ *assumption states that for any* PPT Samp *and* $\mathcal{B}$*, there exists a* PPT $\mathcal{A}$*, such that*

$$\mathsf{Adv}^{\mathsf{Pre}}_{\mathcal{A}}(\lambda) \geq \mathsf{Adv}^{\mathsf{Post}}_{\mathcal{B}}(\lambda)/\mathsf{poly}(\lambda) - \mathsf{negl}(\lambda).$$

Remark 1. Our definition of public-coin evasive LWE is a slight adaptation of the definition given in [BÜW24]. More concretely, we directly set the underlying ring to be $\mathbb{Z}$ in the definition, and we do not pass the output $(\mathbf{A}, \mathbf{P}, \mathsf{aux})$ of Samp to $\mathcal{A}$ and $\mathcal{B}$, and instead pass only the random coins rand of Samp, which can be used to derive the sampler output. Moreover, we do not consider any hints and assume sampling from uniform distributions, while it was shown in [BÜW24] that if we allow sampling the matrix $\mathbf{B}$ from *arbitrary* distributions, then it can lead to counterexamples. We also emphasize that all the counterexamples presented in [BÜW24, DJM+25, HJL25, AMYY25] do not apply to our regime, i.e., to public-coin evasive LWE with uniformly sampled $\mathbf{B}$.

Definition 4 (Tensor LWE [Wee22]). *Let* $n, m, q, \ell, Q \in \mathbb{N}$ *be parameters and* χ *a Gaussian parameter. For all* $\mathbf{x}_1, \ldots, \mathbf{x}_Q \in \{0,1\}^\ell$*, we have*

$$\left\{\mathbf{A}_{\mathsf{attr}}, \{\mathbf{s}^\top(\mathbf{I}_n \otimes \mathbf{r}_i)(\mathbf{A}_{\mathsf{attr}} + \mathbf{x}_i^\top \otimes \mathbf{G}) + \mathbf{e}_i^\top, \mathbf{r}_i\}_{i\in[Q]}\right\} \approx_c \left\{\mathbf{A}_{\mathsf{attr}}, \{\mathbf{c}_i, \mathbf{r}_i\}_{i\in[Q]}\right\},$$

where $\mathbf{A}_{\mathsf{attr}} \xleftarrow{\$} \mathbb{Z}_q^{n\times \ell m}$, $\mathbf{s} \leftarrow \mathbb{Z}_q^{nm}$, $\mathbf{e}_i \leftarrow \mathcal{D}_{\mathbb{Z}^{\ell m},\mathcal{X}}$, $\mathbf{r}_i \leftarrow \mathcal{D}_{\mathbb{Z}^m,\mathcal{X}}$, $\mathbf{c}_i \leftarrow \mathbb{Z}_q^{\ell m}$.

The parameters $m, q, \mathcal{X}$ are chosen to ensure that the assumption $\mathsf{LWE}_{q,n,\mathcal{X}}$ holds with $\ell, Q = \mathsf{poly}(\lambda)$.

2.4 Attribute-Based Encryption

We recall the definition of Attribute-Based Encryption [SW05]. Let $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ and $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$ be sequences of ciphertext-attribute and key-attribute spaces, and let $R = \{R_\lambda\}_{\lambda\in\mathbb{N}}$ be a sequence of relations where $R_\lambda \colon \mathcal{X}_\lambda \times \mathcal{Y}_\lambda \to \{0,1\}$ for all $\lambda \in \mathbb{N}$.

Definition 5 (Attribute-Based Encryption (ABE)). *An ABE scheme* ABE *for the relation* R *is a tuple of* PPT *algorithms* $\mathsf{ABE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ *of the following form:*

$\mathsf{Setup}(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk})$**:** *On input the security parameter* 1^λ*, this algorithm outputs a pair of master public/secret keys* $(\mathsf{mpk}, \mathsf{msk})$*. We assume that* msk *implicitly contains* mpk*.*

$\mathsf{Enc}(\mathsf{mpk}, x, \mu) \to \mathsf{ct}_x$**:** *On input the master public key* mpk *an attribute* $x \in \mathcal{X}_\lambda$ *and a message* $\mu \in \{0,1\}$*, this algorithm outputs a ciphertext* ct_x*.*

$\mathsf{KeyGen}(\mathsf{msk}, y) \to \mathsf{sk}_y$**:** *On input the master secret key* msk *and an attribute* $y \in \mathcal{Y}_\lambda$*, this algorithm outputs a secret key* sk_y*.*

$\mathsf{Dec}(\mathsf{sk}_y, \mathsf{ct}_x) \to \mu' \vee \bot$**:** *On input a secret keys* sk_y *and a ciphertext* ct_x*, this algorithm outputs an element* $\mu' \in \{0,1\}$ *or* $\bot$*.*

Relations. We recall the definitions of various ABEs for specific relations.

CP-ABE for Circuits. We define ciphertext-policy ABE (CP-ABE) for a circuit class $\{\mathcal{C}_\lambda\}_\lambda$, where $\mathcal{C}_\lambda$ is a set of circuits with bounded depth $d = d(\lambda)$, input length $n = n(\lambda)$ and binary output, by setting $\mathcal{X}_\lambda = \mathcal{C}_\lambda$, $\mathcal{Y}_\lambda = \{0,1\}^n$ and $R_\lambda(C, \mathbf{x}) = C(\mathbf{x})$.

KP-ABE for Circuits. We define key-policy ABE (KP-ABE) for circuits by swapping the key and ciphertext attributes in CP-ABE for circuits. More concretely, we define KP-ABE for circuits by setting $\mathcal{X}_\lambda = \{0,1\}^n$, $\mathcal{Y}_\lambda = \mathcal{C}_\lambda$ and $R_\lambda(\mathbf{x}, C) = C(\mathbf{x})$.

Correctness. The ABE scheme ABE is correct if for all $\lambda \in \mathbb{N}$, $\mu \in \{0,1\}$, $x \in \mathcal{X}_\lambda$ and $y \in \mathcal{Y}_\lambda$ such that $R_\lambda(x,y) = 0$, we have

$$\Pr\left[\mu \neq \mu' \;\middle|\; \begin{array}{l} (\mathsf{mpk},\mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{sk}_y \leftarrow \mathsf{KeyGen}(\mathsf{msk}, y) \\ \mathsf{ct}_x \leftarrow \mathsf{Enc}(\mathsf{mpk}, x, \mu) \\ \mu' := \mathsf{Dec}(\mathsf{sk}_y, \mathsf{ct}_x) \end{array}\right] = \mathsf{negl}(\lambda) \ ,$$

where the probability is taken over the random coins of the algorithms.

Security. We define security for ABE as follows.

Definition 6 (IND-CPA Security). *Let* $\mathsf{xx} \in \{\mathsf{adap}, \mathsf{sel}, \mathsf{vsel}\}$. *Given a* PPT *adversary* $\mathcal{A}$ *against an ABE scheme* ABE *for a predicate* $R = \{R_\lambda\}_{\lambda\in\mathbb{N}}$, *we define the experiment* $\mathsf{Exp}^{\mathsf{abe}\text{-}b}_{\mathsf{ABE},\mathcal{A}}(1^\lambda)$ *as shown in Figure 1. If* $\mathsf{xx} = \mathsf{sel}$, *then the adversary must declare the challenge attribute* x^* *before receiving* mpk. *If* $\mathsf{xx} = \mathsf{vsel}$, *then the adversary must declare the challenge* (x^*, μ_0^*, μ_1^*) *and all queries to* $\mathcal{O}\mathsf{KeyGen}$ *before receiving* mpk. *We say that* ABE *is* xx*-secure if* $\mathsf{Exp}^{\mathsf{abe}\text{-}0}_{\mathsf{ABE},\mathcal{A}}(1^\lambda) \approx_c \mathsf{Exp}^{\mathsf{abe}\text{-}1}_{\mathsf{ABE},\mathcal{A}}(1^\lambda)$.

$\mathsf{Exp}^{\mathsf{abe}\text{-}b}_{\mathsf{ABE},\mathcal{A}}(1^\lambda)$	$\mathcal{O}\mathsf{KeyGen}(y)$
$1: \mathcal{K} := \emptyset;\ (\mathsf{mpk},\mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$	$1: \mathcal{K} := \mathcal{K} \cup \{y\}$
$2: (x^*, \mu_0^*, \mu_1^*) \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{KeyGen}}(\mathsf{mpk})$	$2:$ **return** $\mathsf{sk}_y \leftarrow \mathsf{KeyGen}(\mathsf{msk}, y)$
$3: \mathsf{ct}_{x^*} \leftarrow \mathsf{Enc}(\mathsf{mpk}, x^*, \mu_b^*)$	
$4: b' \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{KeyGen}}(\mathsf{ct}_{x^*})$	
$5:$ **if** $R_\lambda(x^*, y) = 0$ for all $y \in \mathcal{K}$ **then**	
$6:$ **return** $\beta \leftarrow (b' \stackrel{?}{=} b)$	
$7:$ **return** $\beta \xleftarrow{\$} \{0,1\}$	

Fig. 1. Security game $\mathsf{Exp}^{\mathsf{abe}\text{-}b}_{\mathsf{ABE},\mathcal{A}}(1^\lambda)$ for Definition 6.

2.5 Identity-Based Inner-Product Functional Encryption

We give our definition of Identity-Based Inner-Product Functional Encryption that is adapted from [ACGU20, LLW21] for the case of inner products modulo a prime number q. Let $\{q_\lambda\}_{\lambda\in\mathbb{N}}$ and $\{\mathcal{ID}_\lambda\}_{\lambda\in\mathbb{N}}$ be sequences of moduli and identity spaces, respectively, and let $n = n(\lambda)$ be a polynomial.

Definition 7 (Identity-Based Inner-Product Functional Encryption (IB-IPFE)). *An IB-IPFE scheme* iFE *for* $\{q_\lambda\}_{\lambda\in\mathbb{N}}$ *and* $\{\mathcal{ID}_\lambda\}_{\lambda\in\mathbb{N}}$ *is a tuple of* PPT *algorithms* $\mathsf{iFE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ *of the following form:*

$\mathsf{Setup}(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk})$**:** *On input the security parameter* 1^λ, *this algorithm outputs a pair of master public/secret keys* $(\mathsf{mpk}, \mathsf{msk})$. *We assume that* msk *implicitly contains* mpk.

$\mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, \mathbf{x}) \to \mathsf{sk}_{\mathbf{x}}$**:** *On input the master public key* mpk, *an identity* $\mathsf{id} \in \mathcal{ID}_\lambda$ *and a message vector* $\mathbf{x} \in \mathbb{Z}_{q_\lambda}^n$, *this algorithm outputs a ciphertext* $\mathsf{ct}_{\mathsf{id}}$.

$\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}', \mathbf{y}) \to \mathsf{sk}_{\mathsf{id}',\mathbf{y}}$**:** *On input the master secret key* msk, *an identity* $\mathsf{id}' \in \mathcal{ID}_\lambda$ *and a key vector* $\mathbf{y} \in \mathbb{Z}_{q_\lambda}^n$, *this algorithm outputs a secret key* $\mathsf{sk}_{\mathsf{id}',\mathbf{y}}$.

$\mathsf{Dec}(\mathsf{sk}_{\mathsf{id}',\mathbf{y}}, \mathsf{ct}_{\mathsf{id}}) \to \mu \vee \perp$**:** *On input a secret keys* $\mathsf{sk}_{\mathsf{id}',\mathbf{y}}$ *and a ciphertext* $\mathsf{ct}_{\mathsf{id}}$, *this algorithm outputs an element* $\mu \in \mathbb{Z}_{q_\lambda}$ *or* $\perp$.

We call iFE *an IPFE scheme if* $\mathcal{ID}_\lambda = \{\top\}$ *is a singleton for all* $\lambda \in \mathbb{N}$. *For convenience, we may omit the second input to* Enc *and* KeyGen *in this case.*

Correctness. The IB-IPFE scheme iFE is correct if for all $\lambda \in \mathbb{N}$, $\mathsf{id} \in \mathcal{ID}_\lambda$, and $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_{q_\lambda}^n$, we have

$$\Pr\left[\mu \neq \mathbf{x}^\top \mathbf{y} \;\middle|\; \begin{array}{l} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{sk}_{\mathsf{id},\mathbf{y}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}, \mathbf{y}) \\ \mathsf{ct}_{\mathsf{id}} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, \mathbf{x}) \\ \mu := \mathsf{Dec}(\mathsf{sk}_{\mathsf{id},\mathbf{y}}, \mathsf{ct}_{\mathsf{id}}) \end{array}\right] = \mathsf{negl}(\lambda) \ ,$$

where the probability is taken over the random coins of the algorithms.

Security. We define single-challenge SIM-based security for IB-IPFE as follows.

Definition 8 ((q_0, q_1)-SIM Security). *Let* $\mathsf{xx} \in \{\mathsf{adap}, \mathsf{sel}\}$. *Given a* PPT *adversary* $\mathcal{A}$ *against an IB-IPFE scheme* iFE *and a* PPT *simulator* $\mathsf{Sim} = (\mathsf{Setup}^*, \mathsf{Enc}^*, \mathsf{KeyGen}^*_{\mathsf{pre}}, \mathsf{KeyGen}^*_{\mathsf{post}})$, *we consider* $\mathsf{Exp}^{\mathsf{real}}_{\mathsf{iFE},\mathcal{A}}(1^\lambda)$ *and* $\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{Sim},\mathcal{A}}(1^\lambda)$ *as depicted in Figure 2. If* $\mathsf{xx} = \mathsf{sel}$, *then the adversary must declare the challenge identities* $\{\mathsf{id}^*\}$ *before receiving* mpk *or* mpk^*. *We say that* iFE *is* xx-(q_0, q_1)-*SIM secure if* $\mathsf{Exp}^{\mathsf{real}}_{\mathsf{iFE},\mathcal{A}}(1^\lambda) \approx_c \mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{Sim},\mathcal{A}}(1^\lambda)$.

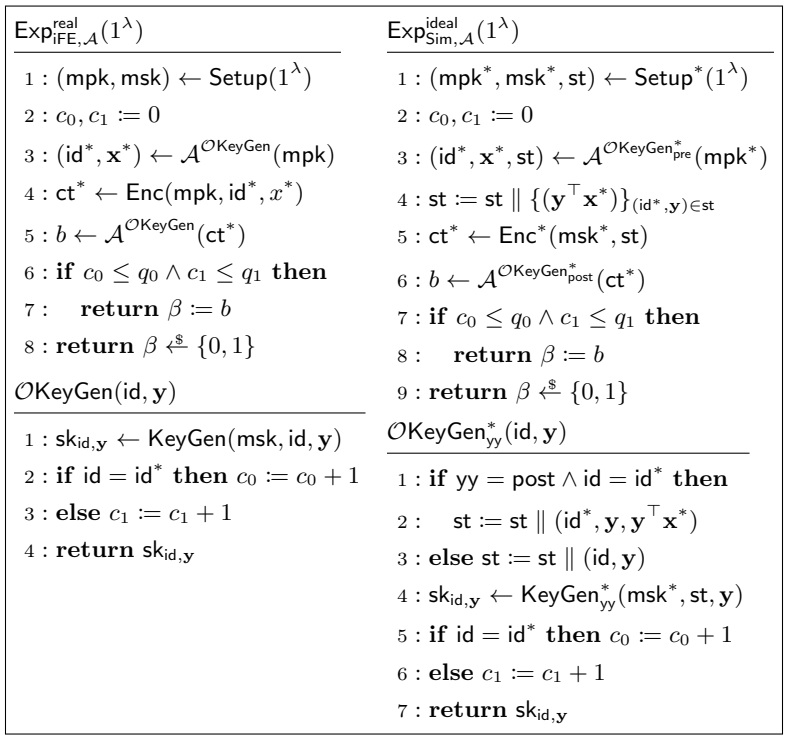


Fig. 2. Security experiments $\mathsf{Exp}^{\mathsf{real}}_{\mathsf{iFE},\mathcal{A}}(1^\lambda)$ and $\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{Sim},\mathcal{A}}(1^\lambda)$ for Definition 8.

For the special case of IPFE, we additionally define multi-challenge SIM-based security against a single key query, i.e., without collusion resistance.

Definition 9 (1-Key q_c-SIM Security). *Given a* PPT *adversary* $\mathcal{A}$ *against a pair of an IPFE scheme* iFE *and a* PPT *simulator* $\mathsf{Sim} = (\mathsf{Setup}^*, \mathsf{Enc}^*_{\mathsf{pre}}, \mathsf{KeyGen}^*, \mathsf{Enc}^*_{\mathsf{post}})$, *we consider* $\mathsf{Exp}^{\mathsf{real}}_{\mathsf{iFE},\mathcal{A}}(1^\lambda)$ *and* $\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{Sim},\mathcal{A}}(1^\lambda)$ *as shown in Figure 3. We say that* iFE *is 1-key* q_c*-challenge SIM secure if* $\mathsf{Exp}^{\mathsf{real}}_{\mathsf{iFE},\mathcal{A}}(1^\lambda) \approx_c \mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{Sim},\mathcal{A}}(1^\lambda)$.

Note that, throughout the experiment $\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{Sim},\mathcal{A}}(1^\lambda)$, *the simulator does not have access to the set* $\mathcal{C}_{\mathsf{pre}}$.

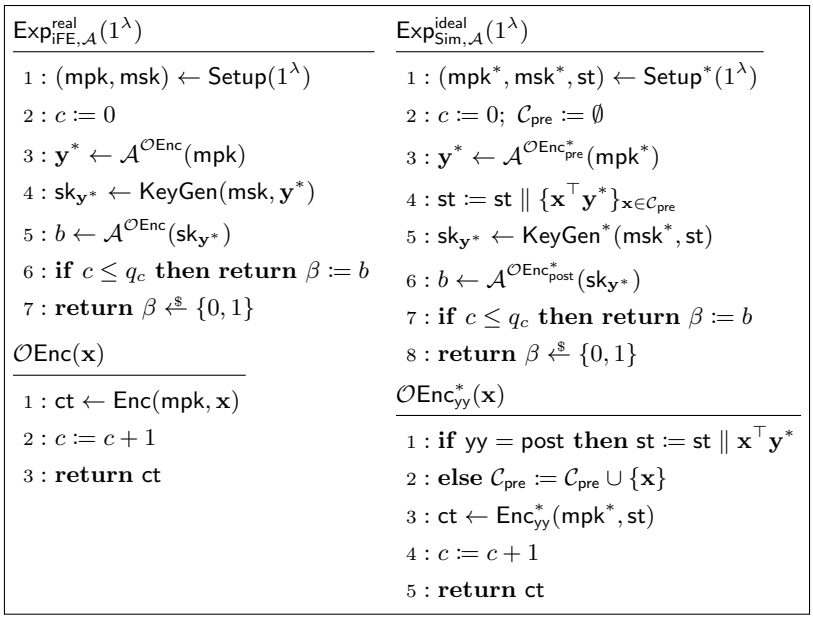


Fig. 3. Security experiments $\mathsf{Exp}^{\mathsf{real}}_{\mathsf{iFE},\mathcal{A}}(1^\lambda)$ and $\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{Sim},\mathcal{A}}(1^\lambda)$ for Definition 9.

2.6 Lockable Obfuscation

We recall the definition of a lockable obfuscator [GKW17, WZ17]. Given polynomials $n = n(\lambda)$, $m = m(\lambda)$ and $d = d(\lambda)$, we denote by $\mathcal{C}_{n,m,d}(\lambda)$ the class of depth $d(\lambda)$ circuits with $n(\lambda)$ bits input and $m(\lambda)$ bits output.

Definition 10 (Lockable Obfuscation). *Consider a sequence of message spaces $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ and a sequence of circuit classes $\{\mathcal{C}_{n,m,d}(\lambda)\}_{\lambda\in\mathbb{N}}$ a sequence of circuit classes. A lockable obfuscator for $\mathcal{M}$ and $\mathcal{C}$ is a tuple of* PPT *algorithms* $(\mathsf{Obf},\mathsf{Eval},\mathsf{Sim})$ *of the following form:*

$\mathsf{Obf}(1^\lambda, C, \mu, \sigma) \to (\widetilde{C})$**:** *On input 1^λ, a circuit $C \in \mathcal{C}_{n,m,d}(\lambda)$, a message $\mu \in \mathcal{M}_\lambda$ and a "lock value" $\sigma \in \{0,1\}^{m(\lambda)}$, this algorithm outputs an obfuscated circuit $\widetilde{C}$.*

$\mathsf{Eval}(\widetilde{C}, x) \to \mu' \vee \bot$**:** *On input an obfuscated circuit $\widetilde{C}$ and an input $x \in \{0,1\}^{n(\lambda)}$, this algorithm outputs a value $\mu' \in \mathcal{M}_\lambda$ or $\bot$.*

$\mathsf{Sim}(1^\lambda, 1^{|C|}, 1^{|\mu|})$**:** *On input 1^λ, the size of a circuit $1^{|C|}$ and the length of a message $1^{|\mu|}$, this algorithm outputs a simulated obfuscation $\widetilde{C}$.*

Correctness. A lockable obfuscator satisfies (perfect) correctness if for all $\lambda \in \mathbb{N}$, all circuits $C \in \mathcal{C}_{n,m,d}(\lambda)$, all messages $\mu \in \mathcal{M}_\lambda$ and all inputs $x \in \{0,1\}^{n(\lambda)}$, the following two implications are satisfied:

1. If $C(x) = \sigma$, then $\mathsf{Eval}(\mathsf{Obf}(1^\lambda, C, \mu, \sigma), x) = \mu$.
2. If $C(x) \neq \sigma$, then $\mathsf{Eval}(\mathsf{Obf}(1^\lambda, C, \mu, \sigma), x) = \bot$.

Security. We define security against multiple challenges. In [AYY22], this definition was observed to be equivalent to the original single-challenge version from [GKW17].

Definition 11 (Security against Multiple Queries). *For a lockable obfuscation scheme* $\mathsf{LO} = (\mathsf{Obf},\mathsf{Eval},\mathsf{Sim})$*, we define the following oracles:*

$\mathcal{O}\mathsf{Obf}_0(C,\mu)$**:** *Sample $\sigma \xleftarrow{\$} \{0,1\}^{m(\lambda)}$ and return $\widetilde{C} \leftarrow \mathsf{Obf}(1^\lambda, C, \mu, \sigma)$.*

$\mathcal{O}\mathsf{Obf}_1(C,\mu)$**:** *Return $\widetilde{C} \leftarrow \mathsf{Sim}(1^\lambda, 1^{|C|}, 1^{|\mu|})$.*

We call LO *secure if for all PPT adversaries* $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}^{\mathsf{lock}}_{\mathsf{LO},\mathcal{A}}(\lambda) := \left|\Pr\left[\mathcal{A}^{\mathcal{O}\mathsf{Obf}_1} \to 1\right] - \Pr\left[\mathcal{A}^{\mathcal{O}\mathsf{Obf}_0} \to 1\right]\right| = \mathsf{negl}(\lambda) \ .$$

Perfectly correct lockable obfuscators for general circuits are known to exist under the LWE assumption [GKW17, GKVW20].

2.7 Other Preliminaries

We recall the following lemma from [LLW21].

Lemma 12 ([LLW21], Lemma 5.4). *Let* $N = Qt\lambda^2$ *and* $t = \Theta(\lambda)$*. There exists an efficient sampler* $\mathsf{SamplerSet}(N, Q, t)$ *with the following properties:*

1. *The sampler always outputs a set* $\Delta \subset [N]$ *with cardinality* t*.*
2. *For independent samples* $\Delta_1, \ldots, \Delta_Q$ *from* $\mathsf{SamplerSet}(N, Q, t)$*, the sets are cover-free with probability* $(1 - 2^{-\Omega(\lambda)})$*, i.e., for all* $i \in [Q]$*,*

$$\Pr\left[\Delta_i \setminus \left(\bigcup_{j\in[Q]\setminus\{i\}} \Delta_j\right) \neq \emptyset\right] \geq 1 - 2Q \cdot 2^{-\Omega(\lambda)} \ .$$

3 Ciphertext-Updatable Attribute-Based and Predicate Encryption

Preliminaries on notation and all necessary building blocks can be found in Section 2. We define ciphertext-updatable attribute-based encryption (CU-ABE) and ciphertext-updatable predicate encryption (CU-PE). Since the only difference between these notions lies in the security game, we unify the syntax of the algorithms. Our definition only covers the case of unidirectional updates, since this is the relevant setting for our constructions.

Let $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ and $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$ be sequences of ciphertext-attribute and key-attribute spaces, and let $R = \{R_\lambda\}_{\lambda\in\mathbb{N}}$ be a sequence of relations where $R_\lambda \colon \mathcal{X}_\lambda \times \mathcal{Y}_\lambda \to \{0,1\}$ for all $\lambda \in \mathbb{N}$.

Definition 12 (Syntax). *An L-hop CU-ABE (resp. CU-PE) scheme* E *for* R *is a tuple of* PPT *algorithms* $\mathsf{E} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{TokGen}, \mathsf{Update}, \mathsf{Dec})$ *of the following form:*

$\mathsf{Setup}(1^\lambda, 1^L) \to (\mathsf{mpk}, \mathsf{msk})$**:** *On input the security parameter* 1^λ *and the maximum number of hops* 1^L *in unary, this algorithm outputs a pair of master public/secret keys* $(\mathsf{mpk}, \mathsf{msk})$*. We assume that* msk *implicitly contains* mpk*.*

$\mathsf{Enc}(\mathsf{mpk}, x, \mu) \to \mathsf{ct}$**:** *On input the master public key* mpk*, a ciphertext attribute* $x \in \mathcal{X}_\lambda$ *and a message* $\mu \in \{0,1\}$*, this algorithm outputs a ciphertext* ct*.*

$\mathsf{KeyGen}(\mathsf{msk}, y) \to \mathsf{sk}_y$**:** *On input the master secret key* msk *and a key attribute* $y \in \mathcal{Y}_\lambda$*, this algorithm outputs a secret key* sk_y*. We assume that* sk_y *implicitly contains* mpk*.*

$\mathsf{TokGen}(\mathsf{msk}, x, x') \to \Delta_{x\to x'}$**:** *On input the master secret key* mpk *and a pair of ciphertext attributes* $x, x' \in \mathcal{X}_\lambda$*, this algorithm outputs an update token* $\Delta_{x\to x'}$*.*

$\mathsf{Update}(\Delta_{x\to x'}, \mathsf{ct}) \to \mathsf{ct}'$**:** *On input an update token* $\Delta_{x\to x'}$ *for attributes* $x, x' \in \mathcal{X}_\lambda$ *and a ciphertext* ct *generated with respect to* x*, this algorithm outputs an updated ciphertext* ct'*.*

$\mathsf{Dec}(\mathsf{sk}_y, \mathsf{ct}) \to \mu'$ **or** $\{\bot\}$**:** *On input a secret key* sk_y *and a ciphertext* ct *created by* Enc *or* Update*, this algorithm outputs a message* $\mu' \in \{0,1\}$ *or* $\bot$*, which represents the rejection symbol.*

L-Correctness. The CU-ABE (resp. CU-PE) scheme E is L-correct if for all $\lambda \in \mathbb{N}$, $\ell \in [0; L]$, $\mu \in \{0,1\}$, $x_0, \ldots, x_\ell \in \mathcal{X}_\lambda$, and $y \in \mathcal{Y}_\lambda$ such that $R_\lambda(x_\ell, y) = 0$, we have

$$\Pr\left[\mu \neq \mu' \;\middle|\; \begin{array}{l} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^L) \\ \mathsf{sk}_y \leftarrow \mathsf{KeyGen}(\mathsf{msk}, y) \\ \forall i \in [\ell] \colon \Delta_{x_{i-1}\to x_i} \leftarrow \mathsf{TokGen}(\mathsf{msk}, x_{i-1}, x_i) \\ \mathsf{ct}_0 \leftarrow \mathsf{Enc}(\mathsf{mpk}, x_0, \mu) \\ \forall i \in [\ell] \colon \mathsf{ct}_i \leftarrow \mathsf{Update}(\Delta_{x_{i-1}\to x_i}, \mathsf{ct}_{i-1}) \\ \mu' := \mathsf{Dec}(\mathsf{sk}_y, \mathsf{ct}_\ell) \end{array}\right] = \mathsf{negl}(\lambda) \ ,$$

where the probability is taken over the random coins of the algorithms.

Security. Our security definition combines the security definitions of ABE (as defined in Definition 6) with ciphertext updatability described in [CRS+23]. Specifically, we want update tokens to satisfy two security requirements:

(1) while a token allows updating the attribute of a ciphertext, it *cannot* be used for decryption, and
(2) seeing an updated ciphertext is *not* sufficient to update other ciphertexts as well. In particular, an updated ciphertext does not disclose the token used during its update procedure.

Let us analyze (2) first. Our security game includes an oracle $\mathcal{O}\mathsf{Update}$ which allows updating arbitrary ciphertexts that were honestly generated (using the encryption oracle $\mathcal{O}\mathsf{Enc}$ or the challenge ciphertext). Therefore, the adversary can call $\mathcal{O}\mathsf{Enc}$ on input the challenge attribute x_b^* to obtain a ciphertext ct, which can subsequently be transformed into a ciphertext ct' with respect to an attribute x' using the oracle $\mathcal{O}\mathsf{Update}$. Assume that the adversary additionally possesses a secret key sk_y for a key attribute y such that $R(x', y) = 0$. Now we can observe that *if* the updated ciphertext ct' also allowed updating other ciphertexts from x_b^* to x', then the security game could be won by updating the attribute of the challenge ciphertext to x' followed by a decryption using sk_y. Hence, to satisfy our security definition, updated ciphertexts cannot leak information about the token that was used during their update procedure.

To guarantee (1), our definition essentially allows the adversary to obtain arbitrary tokens $\Delta_{x\to x'}$ by querying the oracle $\mathcal{O}\mathsf{TokGen}(x, x')$. In particular, these tokens may be used to update the challenge ciphertext. For the definition to be meaningful, the admissibility condition forbids the adversary to obtain a set of tokens $\{\Delta_{x_{i-1}\to x_i}\}_{i\in[\ell]}$ along with a secret key sk_y such that $R(x_\ell, y) = 0$, as it would enable a chain of legitimate update operations on the challenge ciphertext eventually yielding a ciphertext that can be decrypted using sk_y. Note that for compatibility with (2), the admissibility condition must forbid such sets of update chains not only for the challenge ciphertext itself but also for updates of the challenge ciphertext obtained from $\mathcal{O}\mathsf{Update}$.

Definition 13 (IND-CPA Security). *Let* $\mathsf{xx} \in \{CU\text{-}ABE, CU\text{-}PE\}$ *and* $\mathsf{yy} \in \{\mathsf{adap}, \mathsf{sel}, \mathsf{vsel}\}$. *Given a* PPT *adversary* $\mathcal{A}$ *against an* xx *scheme* E *for a predicate* $R = \{R_\lambda\}_{\lambda\in\mathbb{N}}$, *we define the experiment* $\mathsf{Exp}^{\mathsf{cu\text{-}pe\text{-}}b}_{\mathsf{E},\mathcal{A}}(1^\lambda)$ *as shown in Figure 4, where* $\mathcal{O} := (\mathcal{O}\mathsf{KeyGen}, \mathcal{O}\mathsf{Enc}, \mathcal{O}\mathsf{TokGen}, \mathcal{O}\mathsf{Update})$ *denotes the collection of oracles that the adversary has access to. Let* $\mathcal{K}$, $\mathcal{C}$, $\mathcal{D}$, $\mathcal{CT}$ *and* $\mathcal{HT}$ *be the sets of secret key queries, (honest) ciphertext queries, challenge ciphertext indices, corrupted tokens and honest tokens, respectively. The adversary* $\mathcal{A}$ *is admissible with respect to* $\mathcal{K}$, $\mathcal{C}$, $\mathcal{D}$, $\mathcal{CT}$ *and* $\mathcal{HT}$, *denoted by* $\mathsf{adm}(\mathcal{A}) = 0$, *if the following conditions are satisfied. Otherwise, we say that* $\mathcal{A}$ *is not admissible and write* $\mathsf{adm}(\mathcal{A}) = 1$.

1. *For each* $b \in \{0,1\}$, $(\cdot, x_0', x_1') \in \mathcal{D}$ *and* $y \in \mathcal{K}$, *it holds that* $R_\lambda(x_b', y) = 1$, i.e. *the adversary did not query* $\mathcal{O}\mathsf{KeyGen}$ *on any secret key that would enable a legitimate decryption of the challenge ciphertext or any of its updates.*
2. *For each* $b \in \{0,1\}$, $(\cdot, x_0', x_1') \in \mathcal{D}$, $y \in \mathcal{K}$ *and* $\{(x_{i-1}, x_i)\}_{i\in[\ell]} \subseteq \mathcal{CT}$ *such that* $x_0 = x_b'$, *it holds that* $R_\lambda(x_\ell, y) = 1$, *i.e., the adversary has not received a secret key together with a chain of update tokens that would enable a legitimate update-then-decrypt procedure for the challenge ciphertext.*
3. *If* $\mathsf{xx} = CU\text{-}ABE$, *then* $x_0^* = x_1^*$ *and* $x_0' = x_1'$ *for each query* $\mathcal{O}\mathsf{Update}(i, x_0', x_1')$.
4. *If* $\mathsf{yy} = \mathsf{sel}$, *then the adversary must declare the challenge attributes* (x_0^*, x_1^*) *before receiving* mpk.
5. *If* $\mathsf{yy} = \mathsf{vsel}$, *then the adversary must declare the challenge* $(x_0^*, x_1^*, \mu_0^*, \mu_1^*)$ *and all queries to* $\mathcal{O} := (\mathcal{O}\mathsf{KeyGen}, \mathcal{O}\mathsf{TokGen}, \mathcal{O}\mathsf{Update})$ *before receiving* mpk *and has access only to the following oracles* $\mathcal{O}\mathsf{Enc}$.

We say that E *is* yy*-secure if* $\mathsf{Exp}^{\mathsf{cu\text{-}pe\text{-}0}}_{\mathsf{E},\mathcal{A}}(1^\lambda) \approx_c \mathsf{Exp}^{\mathsf{cu\text{-}pe\text{-}1}}_{\mathsf{E},\mathcal{A}}(1^\lambda)$.

In the case of CU-ABE, we may simplify notations by only considering single attributes x^*, x' instead of pairs $(c_0^*, x_1^*), (x_0', x_1')$.

4 Single-Hop Ciphertext-Updatable ABE and PE from LWE

In this section, we present two generic compilers to obtain single-hop CU-ABE and CU-PE for a bounded number of tokens. This approach allows us to attain schemes based on plain LWE.

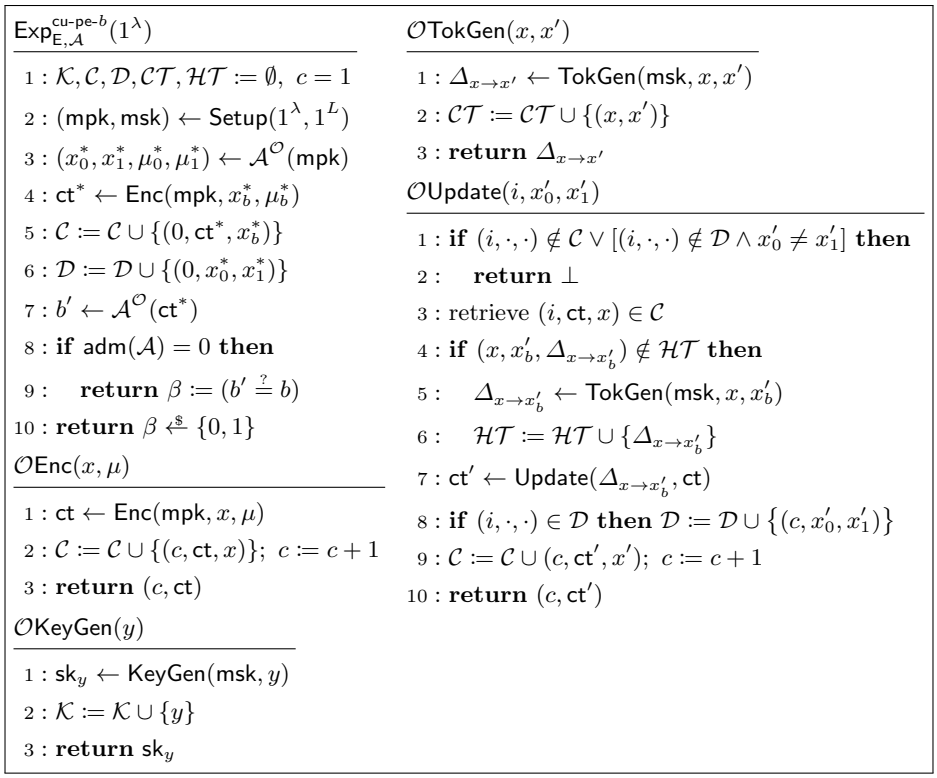


Fig. 4. Security game $\mathsf{Exp}^{\mathsf{cu\text{-}pe\text{-}}b}_{\mathsf{E},\mathcal{A}}(1^\lambda)$ for Definition 13.

4.1 Ciphertext-Updatable ABE

Our first generic compiler converts a traditional ABE into a single-hop CU-ABE. Intuitively, it can be applied to any ABE scheme where the encryption algorithm $\mathsf{Enc}(\mathsf{mpk}, x, \mu; \mathbf{r})$ can be divided into an offline and an online phase such that

- the offline phase draws a sample $\mathbf{u}$ from a fixed distribution given only the master public key mpk and the random coins $\mathbf{r}$, and
- the deterministic online phase computes a *linear* function of the message input μ and the sample $\mathbf{u}$.

We formalize this property in the following definition.

Definition 14 (Linearly Constructible Ciphertexts). *An ABE scheme* $\mathsf{ABE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *for a sequence of relations* $R = \{R_\lambda : \mathcal{X}_\lambda \times \mathcal{Y}_\lambda \to \{0,1\}\}_{\lambda \in \mathbb{N}}$ *is said to have* linearly constructible ciphertexts *if there exists a pair of efficient algorithms* $(\mathsf{GenU}, \mathsf{GenV})$ *with the following properties.*

- Syntax. *There exists* $d = d(\lambda), \ell = \ell(\lambda)$ *such that the algorithms have the following syntax.*

  $\mathsf{GenU}(\mathsf{mpk})$**:** *On input an* ABE *master public key* mpk*, this algorithm outputs a vector* $\mathbf{u} \in \mathbb{Z}_q^{d-1}$.

  $\mathsf{GenV}(\mathsf{mpk}, x)$**:** *On input an* ABE *master public key* mpk *and* $x \in \mathcal{X}_\lambda$*, this algorithm outputs a set of vectors* $\{\mathbf{v}_i\}_{i \in [\ell]}$ *with* $\mathbf{v}_1, \ldots, \mathbf{v}_\ell \in \mathbb{Z}_q^d$ *and some additional information* aux_x*. We require* GenV *to be deterministic.*

– t-Correctness. *For all* $\lambda \in \mathbb{N}$, $\mu \in \{0,1\}$, $x \in \mathcal{X}_\lambda$ *and* $y \in \mathcal{Y}_\lambda$ *such that* $R_\lambda(x,y) = 0$, *it holds that*

$$\Pr\left[\mu' \neq \mu \;\middle|\; \begin{array}{l} (\mathsf{mpk},\mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{sk}_y \leftarrow \mathsf{KeyGen}(\mathsf{msk}, y) \\ \forall j \in [t] \colon \mathbf{u}_j \leftarrow \mathsf{GenU}(\mathsf{mpk}) \\ (\{\mathbf{v}_i\}_{i\in[\ell]}, \mathsf{aux}_x) \leftarrow \mathsf{GenV}(\mathsf{mpk}, x) \\ \mu' \leftarrow \mathsf{Dec}(\mathsf{sk}_y, (\{(\mu \parallel \sum_{j\in[t]} \mathbf{u}_j^\top) \cdot \mathbf{v}_i\}_{i\in[\ell]}, \mathsf{aux}_x)) \end{array}\right] = \mathsf{negl}(\lambda) \ ,$$

where the probability is taken over the coins of Setup, KeyGen *and* GenU.

– Security. *For all* $\lambda \in \mathbb{N}$, $x \in \mathcal{X}_\lambda$ *and* $\mu \in \{0,1\}$, *it holds that* $D_0 \equiv D_1$, *where*

$$D_0 = \left\{ \mathsf{ct}_x \;\middle|\; \begin{array}{l} (\mathsf{mpk},\mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{ct}_x \leftarrow \mathsf{Enc}(\mathsf{mpk}, x, \mu) \end{array} \right\}$$

$$D_1 = \left\{ (\{(\mu \parallel \mathbf{u}^\top) \cdot \mathbf{v}_i\}_{i\in[\ell]}, \mathsf{aux}_x) \;\middle|\; \begin{array}{l} (\mathsf{mpk},\mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathbf{u} \leftarrow \mathsf{GenU}(\mathsf{mpk}) \\ (\{\mathbf{v}_i\}_{i\in[\ell]}, \mathsf{aux}_x) \leftarrow \mathsf{GenV}(\mathsf{mpk}, x) \end{array} \right\}$$

and the probability is over the random coins of Setup, Enc *and* GenU.

The additional information aux_x may be x itself or some encoding of x that is used to decrypt the ciphertext. We may assume that they are also contained in the ABE ciphertext in a public form.

Lemma 13. *The KP-ABE scheme for all bounded-depth arithmetic circuits given in [BGG+14] and the KP-ABE scheme for RAM circuits given in [DHM+24] have linearly constructible ciphertexts for an appropriate choice of parameters.*

Lemma 14. *The CP-ABE scheme for inner-product predicates in [KNYY20] has linearly constructible ciphertexts for an appropriate choice of parameters.*

The proofs can be found in Appendices B.1 and B.2, respectively. We now present our construction.

Construction 1 (Single-Hop Updatable ABE). *Our construction uses the following ingredients:*

– *a* t*-correct ABE scheme* $\mathsf{ABE} = (\mathsf{aSetup}, \mathsf{aEnc}, \mathsf{aKeyGen}, \mathsf{aDec})$ *with linearly constructible ciphertexts,*
– *an IB-IPFE scheme* $\mathsf{iFE} = (\mathsf{iSetup}, \mathsf{iEnc}, \mathsf{iKeyGen}, \mathsf{iDec})$.

The CU-ABE scheme uABE *for* Q *tokens works as follows, where we choose* $N = N(Q)$ *and* $t = t(Q)$ *according to Lemma 12.*

$\mathsf{Setup}(1^\lambda)$ *takes as input the security parameter* 1^λ, *runs* $(\mathsf{aMPK}, \mathsf{aMSK}) \leftarrow \mathsf{aSetup}(1^\lambda)$, $(\mathsf{iMPK}, \mathsf{iMSK}) \leftarrow \mathsf{iSetup}(1^\lambda)$ *and outputs* $\mathsf{MPK} := (\mathsf{aMPK}, \mathsf{iMPK})$ *and* $\mathsf{MSK} := (\mathsf{aMSK}, \mathsf{iMSK})$.

$\mathsf{Enc}(\mathsf{MPK}, x, \mu)$ *takes as input* $\mathsf{MPK} = (\mathsf{aMPK}, \mathsf{iMPK})$, *an attribute* $x \in \mathcal{X}$ *and a message* $\mu \in \{0,1\}$ *and outputs* $\mathsf{CT}_x := (\mathsf{aCT}_x, \mathsf{iCT}_x)$ *as follows:*

$$\begin{aligned} \mathsf{aCT}_x &\leftarrow \mathsf{aEnc}(\mathsf{aMPK}, x, \mu) \\ \mathsf{iCT}_x &\leftarrow \mathsf{iEnc}(\mathsf{iMPK}, x, (\mu \parallel \mathbf{u}_1^\top \parallel \cdots \parallel \mathbf{u}_N^\top)) \ , \end{aligned}$$

where $\mathbf{u}_j \leftarrow \mathsf{GenU}(\mathsf{mpk})$ *for* $j \in [N]$.

$\mathsf{KeyGen}(\mathsf{MSK}, y)$ *takes as input* $\mathsf{MSK} = (\mathsf{aMSK}, \mathsf{iMSK})$ *and* $y \in \mathcal{Y}$ *and outputs* $\mathsf{SK}_y := \mathsf{aSK}_y$ *computed as* $\mathsf{aSK}_y \leftarrow \mathsf{aKeyGen}(\mathsf{aMSK}, y)$.

$\mathsf{TokGen}(\mathsf{MSK}, x, x')$ *takes as input* $\mathsf{MSK} = (\mathsf{aMSK}, \mathsf{iMSK})$ *and a pair of attributes* $(x, x') \in \mathcal{X} \times \mathcal{X}$*, samples a random subset* $S \subseteq [N]$ *of size* t*, computes* $(\{\mathbf{v}_i\}_{i\in[\ell]}, \mathsf{aux}_{x'}) \leftarrow \mathsf{GenV}(\mathsf{mpk}, x')$ *and parses* $\mathbf{v}_i = (v_{i,1}, \mathbf{v}_{i,\geq 2})$*. For each* $i \in \ell$ *and* $j \in [N]$*, the algorithm defines*

$$\mathbf{v}'_{i,j} = \begin{cases} \mathbf{v}_{i,\geq 2} & \text{if } j \in S \\ \mathbf{0} & \text{if } j \in [N] \setminus S \end{cases}$$

and outputs $\Delta_{x\to x'} := (\{\mathsf{iSK}_{x,i}\}_{i\in[\ell]}, \mathsf{aux}_{x'})$*, where* $\mathsf{iSK}_{x,i}$ *is computed as* $\mathsf{iSK}_{x,i} \leftarrow \mathsf{iKeyGen}(\mathsf{iMSK}, x, (v_{i,1} \parallel \mathbf{v}'_{i,1} \parallel \cdots \parallel \mathbf{v}'_{i,N}))$.

$\mathsf{Update}(\mathsf{CT}_x, \Delta_{x\to x'})$ *takes as input a ciphertext* $\mathsf{CT}_x = (\mathsf{aCT}_x, \mathsf{iCT}_x)$ *and a token* $\Delta_{x\to x'} = (\{\mathsf{iSK}_{x,i}\}_{i\in[\ell]}, \mathsf{aux}_{x'})$*, runs* $\mathsf{aCT}_{x',i} \leftarrow \mathsf{iDec}(\mathsf{iSK}_{x,i}, \mathsf{iCT}_x)$ *for* $i \in [\ell]$ *and outputs* $\mathsf{uCT}_{x'} := \mathsf{aCT}_{x'} = (\{\mathsf{aCT}_{x',i}\}_{i\in[\ell]}, \mathsf{aux}_{x'})$.

$\mathsf{Dec}(\mathsf{SK}_y, \mathsf{CT}_x)$ *takes as input a secret key* $\mathsf{SK}_y = \mathsf{aSK}_y$ *and a ciphertext* $\mathsf{CT}_x = (\mathsf{aCT}_x, \mathsf{iCT}_x)$ *(or an updated ciphertext* $\mathsf{CT}_x = \mathsf{aCT}_x$*) and outputs* $\mu' \leftarrow \mathsf{aDec}(\mathsf{aSK}_y, \mathsf{aCT}_x)$.

Correctness and Security. We show that our compiler preserves the security level of the employed ABE scheme and is correct with regard to the t-correctness and correctness of the IB-IPFE scheme. .

Lemma 15. *If* $\mathsf{ABE} = (\mathsf{aSetup}, \mathsf{aEnc}, \mathsf{aKeyGen}, \mathsf{aDec})$ *is a t-correct ABE scheme with linearly constructable ciphertexts and* $\mathsf{iFE} = (\mathsf{iSetup}, \mathsf{iEnc}, \mathsf{iKeyGen}, \mathsf{iDec})$ *is a correct IB-IPFE scheme, then Construction 1 is correct.*

Proof. For a non-updated ciphertext, we just have the regular ABE scheme. Thus, the correctness follows from ABE.

An updated ciphertext $\mathsf{aCT}_{x'}$ is generated using the IB-IPFE scheme, i.e.,

$$\begin{aligned} \mathsf{aCT}_{x',i} &= \langle (v_{i,1} \parallel \mathbf{v}'_{i,1} \parallel \cdots \parallel \mathbf{v}'_{i,N}), (\mu \parallel \mathbf{u}_1^\top \parallel \cdots \parallel \mathbf{u}_N^\top) \rangle \\ &= \mu \cdot v_{i,1} + \sum_{j\in[S]} \mathbf{u}_j^\top \mathbf{v}_{i,\geq 2}, \end{aligned}$$

for some $S \subset [N]$ of size $|S| = t$ by the correctness of iFE. Hence,

$$(\{\mathsf{aCT}_{x',i}\}_{i\in[\ell]}, \mathsf{aux}_{x'}) = (\{(\mu \parallel \sum_{j\in[S]} \mathbf{u}_j^\top) \cdot \mathbf{v}_i\}_{i\in[\ell]}, \mathsf{aux}_{x'}),$$

which is decrypted correctly with high probability by the t-Correctness of ABE. □

Theorem 2. *Let* $\mathsf{xx} \in \{\mathsf{adap}, \mathsf{sel}\}$*. If* ABE *is* xx*-IND-secure and* iFE *is* (Q, poly)*-*xx*-SIM-secure, then the CU-ABE scheme* uABE *is* xx*-IND-secure if the adversary submits queries to* $\mathcal{O}\mathsf{TokGen}$ *and* $\mathcal{O}\mathsf{Update}$ *for at most* Q *different pairs of attributes* $(x, x') \in \mathcal{X} \times \mathcal{X}$*, i.e.,* $|\mathcal{CT}| + |\mathcal{HT}| \leq Q$ *in Figure 4.*

Proof. The proof consists of a sequence of hybrid games $\mathsf{G}_0^b, \mathsf{G}_1^b, \mathsf{G}_2^b, \mathsf{G}_3^b$, for $b \in \{0, 1\}$. Modifications between consecutive games are highlighted using boxes.

G_0^b **for** $b \in \{0, 1\}$**:** This is the original game with challenge bit b.

G_1^b **for** $b \in \{0, 1\}$**:** We first deal with the ciphertext component aCT_{x^*}. G_1^b is the same as G_0^b except that the challenge ciphertext is of the form $\mathsf{CT}_{x^*} = (\mathsf{aCT}_{x^*}, \mathsf{iCT}_{x^*})$, where

$$\begin{aligned} \mathsf{aCT}_{x^*} &\leftarrow \mathsf{aEnc}(\mathsf{aMPK}, x^*, \boxed{\mu_0^*}) \\ \mathsf{iCT}_{x^*} &\leftarrow \mathsf{iEnc}(\mathsf{iMPK}, x^*, (\mu_b^* \parallel \mathbf{u}_1^\top \parallel \cdots \parallel \mathbf{u}_N^\top)) \ . \end{aligned}$$

We have $\mathsf{G}_0^b \approx_c \mathsf{G}_1^b$ which follows from the security of ABE. The rest of the proof deals with iCT_{x^*}.

G_2^b **for** $b \in \{0, 1\}$**:** This is the same as G_1^b except that the challenger uses the simulator for iFE.

– The master public key is of the form $\mathsf{MPK} \coloneqq (\mathsf{aMPK}, \mathsf{iMPK}^*)$, where

$$(\mathsf{aMPK}, \mathsf{aMSK}) \leftarrow \mathsf{aSetup}(1^\lambda)$$
$$(\mathsf{iMPK}^*, \mathsf{iMSK}^*, \mathsf{st}) \leftarrow \mathsf{iSetup}^*(1^\lambda)$$

– Update tokens are of the form $\Delta_{x \to x'} \coloneqq \{\mathsf{iSK}_{x,i}\}_{i \in [\ell]}$, where

$$\mathsf{iSK}_{x,i} \leftarrow \mathsf{iKeyGen}^*(\mathsf{iMSK}^*, x, (v_{i,1} \parallel \mathbf{v}'_{i,1} \parallel \cdots \parallel \mathbf{v}'_{i,N}), \mathsf{st}_b)$$

and $\{(v_{i,1} \parallel \mathbf{v}'_{i,1} \parallel \cdots \parallel \mathbf{v}'_{i,N})\}_{i \in [\ell]}$ is generated as before.

– The challenge ciphertext is of the form $\mathsf{CT}_{x^*} \coloneqq (\mathsf{aCT}_{x^*}, \mathsf{iCT}^*_{x^*})$, where

$$\mathsf{aCT}_{x^*} \leftarrow \mathsf{aEnc}(\mathsf{aMPK}, x^*, \mu)$$
$$\mathsf{iCT}^*_{x^*} \leftarrow \mathsf{iEnc}^*(\mathsf{iMSK}^*, x^*, \mathsf{st}_b) \quad ,$$

and st_b contains the inner products between the IB-IPFE message vectors $(\mu^*_b \parallel \mathbf{u}_1^\top \parallel \cdots \parallel \mathbf{u}_N^\top)$ and the IB-IPFE key vectors $\{(v_{i,1} \parallel \mathbf{v}'_{i,1} \parallel \cdots \parallel \mathbf{v}^*_{i,N})\}_{i \in [\ell]}$ for all previously queried tokens.

G_3^b **for** $b \in \{0,1\}$**:** This is the same as G_2^b except that the challenger samples the Q size t subsets $S_1, \ldots, S_Q \subseteq [N]$ upfront and picks an index $j_q \in S_q$ for each $q \in [Q]$ such that $j_q \notin \bigcup_{q' \in [Q] \setminus \{q\}} S_{q'}$. If such indices do not exist for each $q \in [Q]$, then the challenger aborts. During the generation of the q-th update token (note that this may happen while computing the response to either an $\mathcal{O}\mathsf{TokGen}$ or an $\mathcal{O}\mathsf{Update}$ query), the challenger uses S_q instead of sampling a fresh set S. By Lemma 12, the probability of abortion is negligible which implies that $\mathsf{G}_2^b \approx_s \mathsf{G}_3^b$.
Furthermore, we have that $\mathsf{G}_3^0 \approx_c \mathsf{G}_3^1$ which can be seen as follows. For $q \in [Q]$, let (x_q, x'_q) denote the pair of attributes with respect to which the challenger generates the q-th update token $\Delta_{x_q \to x'_q} = \{\mathsf{iSK}_{x_q,i}\}_{i \in [\ell]}$. If $x_q \neq x^*$, then there is nothing to show since iFE is identity-based and nothing is revealed in this case. Otherwise, decryption of iCT^* with $\{\mathsf{iSK}_{x_q,i}\}_{i \in [\ell]}$ yields $\{\alpha_{i,b}\}_{i \in [\ell]}$ for

$$\alpha_{i,b} \coloneqq \mu^*_b v_{i,1} + \sum_{j \in [N]} \mathbf{u}_j^\top \mathbf{v}_{i,j} = \mu^*_b v_{i,1} + \sum_{j \in S_q} \mathbf{u}_j^\top \mathbf{v}_{i,\geq 2} \ ,$$

where $\mathbf{u}_j \leftarrow \mathsf{GenU}(\mathsf{aMPK})$ for $j \in [N]$, $(\{\mathbf{v}_i = (v_{i,1}, \mathbf{v}_{i,\geq 2})\}_{i \in [\ell]}, \mathsf{aux}_{x'_q}) \leftarrow \mathsf{GenV}(\mathsf{aMPK}, x'_q)$ and

$$\mathbf{v}'_{i,j} = \begin{cases} \mathbf{v}_{i,\geq 2} & \text{if } j \in S_q \\ \mathbf{0} & \text{if } j \in [N] \setminus S_q \ . \end{cases}$$

From the definition of linearly constructible ciphertexts and the fact that j_q is not contained in any $S_{q'}$ for $q' \neq q$, it follows that $(\{(\mu^*_b \parallel \mathbf{u}_{j_q}^\top) \cdot \mathbf{v}_i\}_{i \in [\ell]}, \mathsf{aux}_{x'_q})$ is identically distributed as a fresh ciphertext $\mathsf{ct}_{x'_q} \leftarrow \mathsf{aEnc}(\mathsf{aMPK}, x'_q, \mu^*_b)$. Parse $\mathsf{ct}_{x'_q} = (\{c_{b,i}\}_{i \in [\ell]}, \mathsf{aux}_{x'_q})$. Then we can write

$$\alpha_{i,b} = c_{b,i} + \sum_{j \in S_q \setminus \{j_q\}} \mathbf{u}_j^\top \mathbf{v}_{i,\geq 2}$$

and conclude that the following distributions are indistinguishable:

$$\begin{aligned} (\{\alpha_{i,0}\}_{i \in [\ell]}, \mathsf{aux}_{x'_q}) &\equiv \left(\{c_{0,i} + \sum_{j \in S_q \setminus \{j_q\}} \mathbf{u}_j^\top \mathbf{v}_{i,\geq 2}\}_{i \in [\ell]}, \mathsf{aux}_{x'_q}\right) \\ &\approx_c \left(\{c_{1,i} + \sum_{j \in S_q \setminus \{j_q\}} \mathbf{u}_j^\top \mathbf{v}_{i,\geq 2}\}_{i \in [\ell]}, \mathsf{aux}_{x'_q}\right) \\ &\equiv (\{\alpha_{i,1}\}_{i \in [\ell]}, \mathsf{aux}_{x'_q}) \ , \end{aligned}$$

where the computational indistinguishability $\approx_c$ follows from the security of ABE. This completes the proof.

□

By combining Lemma 13, Lemma 14 and Theorem 2, we obtain the following.

Corollary 1. *Assuming* LWE*, there exists*

- *a* sel*-secure ciphertext-updatable KP-ABE for all bounded-depth arithmetic circuits,*
- *a* sel*-secure ciphertext-updatable KP-ABE for RAM circuits, and*
- *an* adap*-secure ciphertext-updatable CP-ABE for inner-product predicates.*

4.2 Ciphertext-Updatable Predicate Encryption

In the spirit of linearly constructible "fresh" ciphertexts (Definition 14), we now define linearly constructible updated ciphertexts. Intuitively, we require that fresh ciphertexts (generated by running the Enc algorithm) and update tokens can each be viewed as a set of vectors, and the update process computes the inner products between the two sets.

Definition 15 (Linearly Constructible Updates). *A CU-ABE scheme* $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{TokGen}, \mathsf{Update}, \mathsf{Dec})$ *for a sequence of relations* $R = \{R_\lambda \colon \mathcal{X}_\lambda \times \mathcal{Y}_\lambda \to \{0,1\}\}_{\lambda \in \mathbb{N}}$ *is said to have linearly constructible updates if for all* $\lambda \in \mathbb{N}$*,* $\mu \in \{0,1\}$*,* $x, x' \in \mathcal{X}_\lambda$ *and* $y \in \mathcal{Y}_\lambda$*, it holds that* $D_0 \equiv D_1$*, where*

$$D_0 = \left\{ \mathsf{uct} \;\middle|\; \begin{array}{l} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, x, \mu) \\ \Delta_{x \to x'} \leftarrow \mathsf{TokGen}(\mathsf{msk}, x, x') \\ \mathsf{uct} \leftarrow \mathsf{Update}(\Delta_{x \to x'}, \mathsf{ct}) \end{array} \right\},$$

$$D_1 = \left\{ \mathsf{uct} \;\middle|\; \begin{array}{l} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ \{\mathbf{u}_i\}_{i \in [\ell]} \leftarrow \mathsf{GenU}(\mathsf{mpk}) \\ \{\mathbf{v}_i\}_{i \in [\ell]} \leftarrow \mathsf{GenV}(\mathsf{mpk}, x') \\ \mathsf{uct} \coloneqq \{\mathbf{u}_i^\top \cdot \mathbf{v}_i\}_{i \in [\ell]} \end{array} \right\}.$$

Construction 2 (Single-hop Updatable Predicate Encryption). *Our construction uses the following ingredients:*

- *a CU-ABE scheme* $\mathsf{uABE} = (\mathsf{uSetup}, \mathsf{uEnc}, \mathsf{uKeyGen}, \mathsf{uTokGen}, \mathsf{uUpdate}, \mathsf{uDec})$ *with linearly constructible updates for a message space* $\mathcal{L} = \{0,1\}^m$ *for some* $m = \mathsf{poly}(\lambda)$*, ciphertext attribute universe* $\mathcal{X}$ *and key attribute universe* $\mathcal{Y}$*,*
- *an IPFE scheme* $\mathsf{iFE} = (\mathsf{iSetup}, \mathsf{iEnc}, \mathsf{iKeyGen}, \mathsf{iDec})$*, and*
- *a lockable obfuscation scheme* $\mathsf{LO} = (\mathsf{Obf}, \mathsf{Eval}, \mathsf{Sim})$ *with lock space* $\mathcal{L}$ *and message space* $\mathcal{M}$*. We note that the algorithm* Sim *is only used during the security proof.*

We write $C[x](y)$ *to denote a circuit* C *with* x *hardwired into its description and input* y*. The single-hop CU-PE scheme* uPE *for message space* $\mathcal{M}$*, ciphertext attribute universe* $\mathcal{X}$ *and key attribute universe* $\mathcal{Y}$ *works as follows:*

$\mathsf{Setup}(1^\lambda)$ *takes as input the security parameter* 1^λ*, runs* $(\mathsf{uMPK}, \mathsf{uMSK}) \leftarrow \mathsf{uSetup}(1^\lambda)$ *and outputs* $(\mathsf{MPK} = \mathsf{uMPK}, \mathsf{MSK} = \mathsf{uMSK})$*.*

$\mathsf{Enc}(\mathsf{MPK}, x, \mu)$ *takes as input* $\mathsf{MPK} = \mathsf{uMPK}$*, an attribute* $x \in \mathcal{X}$ *and a message* $\mu \in \mathcal{M}$*. The algorithm samples* $\sigma \xleftarrow{\$} \mathcal{L}$*, runs*

$$\begin{aligned} \{(\mathsf{iMSK}_i, \mathsf{iMPK}_i) &\leftarrow \mathsf{iSetup}(1^\lambda)\}_{i \in [\ell]} \\ \mathsf{uCT}_x = \{\mathbf{u}_{x,i}\}_{i \in [\ell]} &\leftarrow \mathsf{uEnc}(\mathsf{uMPK}, x, \sigma) \\ \{\mathsf{iSK}_{x,i} &\leftarrow \mathsf{iKeyGen}(\mathsf{iMSK}_i, \mathbf{u}_{x,i})\}_{i \in [\ell]} \\ \widetilde{C} &\leftarrow \mathsf{Obf}(1^\lambda, C[\mathsf{uCT}_x, \{\mathsf{iSK}_{x,i}\}_{i \in [\ell]}], \mu, \sigma) \end{aligned}$$

and outputs $\mathsf{CT} \coloneqq (\widetilde{C}, \{\mathsf{iMPK}_i\}_{i \in [\ell]})$*. The circuit* $C[\mathsf{uCT}_x, \{\mathsf{iSK}_{x,i}\}_{i \in [\ell]}]$ *is described in Figure 5.*

hardwired value :
- a uABE ciphertext uCT_x
- a set of iFE secret key $\{\mathsf{iSK}_{x,i}\}_{i\in[\ell]}$

input :
- a uABE secret key uSK_y
- a set of iFE ciphertexts $c = \{\mathsf{iCT}_{x,i}\}_{i\in[\ell]}$ or $c = \perp$

output : an element of the lock space $\sigma \in \mathcal{L}$ or $\perp$

if $c = \perp$ **then**
  $\mathsf{uCT}'_x \coloneqq \mathsf{uCT}_x$
else
  $\forall i \in [\ell]$: $d_i \leftarrow \mathsf{iDec}(\mathsf{iSK}_{x,i}, \mathsf{iCT}_{x,i})$
  $\mathsf{uCT}'_x \coloneqq \{d_i\}_{i\in[\ell]}$
end
return $\sigma \leftarrow \mathsf{uDec}(\mathsf{uSK}_y, \mathsf{uCT}'_x)$

Fig. 5. Definition of the circuit $C[\mathsf{uCT}_x, \{\mathsf{iSK}_{x,i}\}_{i\in[\ell]}](\mathsf{uSK}_y, c)$.

$\mathsf{KeyGen}(\mathsf{MSK}, y)$ *takes as input* $\mathsf{MSK} = \mathsf{uMSK}$ *and a key attribute* $y \in \mathcal{Y}$*, runs* $\mathsf{uSK}_y \leftarrow \mathsf{uKeyGen}(\mathsf{uMSK}, y)$ *and outputs* $\mathsf{SK}_y \coloneqq \mathsf{uSK}_y$.

$\mathsf{TokGen}(\mathsf{MSK}, x, x')$ *takes as input* $\mathsf{MSK} = \mathsf{uMSK}$ *and two attributes* $x, x' \in \mathcal{X}$ *and outputs* $\Delta_{x\to x'} \leftarrow \mathsf{uTokGen}(\mathsf{uMSK}, x, x')$.

$\mathsf{Update}(\mathsf{CT}_x, \Delta_{x\to x'})$ *takes as input a ciphertext* $\mathsf{CT}_x = (\widetilde{C}, \{\mathsf{iMPK}_i\}_{i\in[\ell]})$ *and an update token* $\Delta_{x\to x'} = \{\mathbf{v}_i\}_{i\in[\ell]}$*, runs* $\{\mathsf{iCT}_{x,i} \leftarrow \mathsf{iEnc}(\mathsf{iMPK}_i, \mathbf{v}_i)\}_{i\in[\ell]}$ *and outputs* $\mathsf{CT}'_x = (\widetilde{C}, \{\mathsf{iCT}_{x,i}\}_{i\in[\ell]})$.

$\mathsf{Dec}(\mathsf{SK}_y, \mathsf{CT}_x)$ *takes as input a secret key* $\mathsf{SK}_y = \mathsf{uSK}_y$ *and a ciphertext* $\mathsf{CT}_x = (\widetilde{C}, \widehat{c} = \{\mathsf{iMPK}_i\}_{i\in[\ell]})$ *or an updated ciphertext* $\mathsf{CT}'_x = (\widetilde{C}, \widehat{c} = \{\mathsf{iCT}_{x,i}\}_{i\in[\ell]})$*. The algorithm defines*

$$c = \begin{cases} \perp & \text{if } \widehat{c} = \{\mathsf{iMPK}_i\}_{i\in[\ell]} \\ \{\mathsf{iCT}_{x,i}\}_{i\in[\ell]} & \text{if } \widehat{c} = \{\mathsf{iCT}_{x,i}\}_{i\in[\ell]} \end{cases}.$$

Then it outputs $\mu' \leftarrow \mathsf{Eval}(\widetilde{C}, (\mathsf{uSK}_y, c))$.

Correctness and Security. The following lemmas state the correctness and the security of the compiler, which preserves the security level of the underlying CU-ABE. .

Lemma 16. *Let* $\mathsf{uABE} = (\mathsf{uSetup}, \mathsf{uEnc}, \mathsf{uKeyGen}, \mathsf{uTokGen}, \mathsf{uUpdate}, \mathsf{uDec})$ *be a CU-ABE with linearly constructable updates,* $\mathsf{iFE} = (\mathsf{iSetup}, \mathsf{iEnc}, \mathsf{iKeyGen}, \mathsf{iDec})$ *be a correct IPFE scheme and* $\mathsf{LO} = (\mathsf{Obf}, \mathsf{Eval}, \mathsf{Sim})$ *be a lockable obfuscation scheme. Then Construction 2 is correct.*

Proof. Let $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ be such that $R(x, y) = 0$. For a non-updated ciphertext $\mathsf{CT} = (\widetilde{C}, \{\mathsf{iMPK}_i\}) \leftarrow \mathsf{Enc}(\mathsf{MPK}, x, \mu)$, correctness follows from the correctness of ABE and LO. Specifically, the evaluation of $\widetilde{C}$ on input $(\mathsf{uSK}_y, \perp)$ runs $\mathsf{uDec}(\mathsf{uSK}_y, \mathsf{uCT}'_x)$ where $\mathsf{uCT}_x \leftarrow \mathsf{uEnc}(\mathsf{uMPK}, x, \sigma)$. Under the correctness of uABE, this decryption process returns the lock value σ that was used for the obfuscation of $\widetilde{C}$. Thus, it follows from the correctness of LO that the obfuscated circuit will unlock and the message μ is revealed.

For updated ciphertexts, we additionally rely on the correctness of iFE and the property of linear constructible updates. Let $x, x' \in \mathcal{X}$ and $y \in \mathcal{Y}$ such that $R(x', y) = 0$. Given an update token $\Delta_{x\to x'} = \{\mathbf{v}_i\}_{i\in[\ell]}$, the update process produces $\{\mathsf{iCT}_{x,i} \leftarrow \mathsf{iEnc}(\mathsf{iMPK}_i, \mathbf{v}_i)\}_{i\in[\ell]}$. Similarly, the encryption algorithm parses $\mathsf{uCT}_x = \{\mathbf{u}_i\}_{i\in[\ell]}$ and generates $\{\mathsf{iSK}_{x,i} \leftarrow \mathsf{iKeyGen}(\mathsf{iMSK}_i, \mathbf{u}_{x,i})\}$. The evaluation of the obfuscated circuit $\widetilde{C}$ runs $\mathsf{iDec}(\mathsf{iSK}_{x_i}, \mathsf{iCT}_{x,i})$ for all $i \in [\ell]$ which, assuming the correctness of iFE, yields the inner products $\{\mathbf{u}_{x,i}^\top \mathbf{v}_i\}_{i\in[\ell]}$. From the linear constructibility of updates, it follows that these inner products are

identically distributed as a uABE updated ciphertext w.r.t. the attribute x'. Then we can rely again on the correctness of uABE and LO to conclude that decryption will return μ following the same argument as in the non-updated case. □

Theorem 3. *Let* $\mathsf{xx} \in \{\mathsf{vsel}, \mathsf{sel}, \mathsf{adap}\}$. *If* uABE *is* xx*-IND-secure,* iFE *is 1-key* Q*-challenge SIM-secure and* LO *is SIM-secure, then Construction 2 is* xx*-IND-secure if the adversary submits at most* Q *queries to* $\mathcal{O}\mathsf{Update}$.

Proof. The proof proceeds via a sequence of hybrids $\mathsf{G}_0^b, \mathsf{G}_1^b, \mathsf{G}_2^b, \mathsf{G}_3^b$ for $b \in \{0,1\}$. Modifications between consecutive games are highlighted using boxes.

G_0^b **for** $b \in \{0,1\}$**:** This is the original experiment for challenge bit b. In particular, the challenge ciphertext CT^* is computed as follows:

$$\begin{aligned}\{(\mathsf{iMSK}_i, \mathsf{iMPK}_i) &\leftarrow \mathsf{iSetup}(1^\lambda)\}_{i\in[\ell]}\\ \mathsf{uCT}_{x_b^*} = \{\mathbf{u}_{x_b^*,i}\}_{i\in[\ell]} &\leftarrow \mathsf{uEnc}(\mathsf{uMPK}, x_b^*, \sigma)\\ \{\mathsf{iSK}_{x_b^*,i} &\leftarrow \mathsf{iKeyGen}(\mathsf{iMSK}_i, \mathbf{u}_{x_b^*,i})\}_{i\in[\ell]}\\ \widetilde{C} &\leftarrow \mathsf{Obf}(1^\lambda, C[\mathsf{uCT}_{x_b^*}, \{\mathsf{iSK}_{x_b^*,i}\}_{i\in[\ell]}], \mu_b^*, \sigma)\ .\end{aligned}$$

Responses to $\mathcal{O}\mathsf{Enc}$ queries are computed similarly.

G_1^b **for** $b \in \{0,1\}$**:** This is the same as G_0^b except that the challenger uses the simulator for iFE when replying to queries $\mathcal{O}\mathsf{Enc}(x,\mu)$ queries. Specifically, a (non-challenge) ciphertext is computed as

$$\begin{aligned}\mathsf{uCT}_x = \{\mathbf{u}_{x,i}\}_{i\in[\ell]} &\leftarrow \mathsf{uEnc}(\mathsf{uMPK}, x, \sigma)\\ \{(\mathsf{iMSK}_i^*, \mathsf{iMPK}_i^*, \mathsf{st}) &\leftarrow \mathsf{iSetup}^*(1^\lambda)\}\\ \{\mathsf{iSK}_{x,i} &\leftarrow \mathsf{iKeyGen}^*(\mathsf{iMSK}_i^*, \mathbf{u}_{x,i}, \mathsf{st})\}_{i\in[\ell]}\\ \widetilde{C} &\leftarrow \mathsf{Obf}(1^\lambda, C[\mathsf{uCT}_x, \{\mathsf{iSK}_{x,i}\}_{i\in[\ell]}], \mu, \sigma)\ .\end{aligned}$$

Furthermore, the $\mathcal{O}\mathsf{Update}$ oracle computes updated ciphertexts $\mathsf{CT}_{x'} = (\widetilde{C}, \{\mathsf{iCT}_{x,i}\}_{i\in[\ell]})$ using a token $\Delta_{x\to x'} = \{\mathbf{v}_i\}_{i\in[\ell]}$ as follows:

$$\{\mathsf{iCT}_{x_i} \leftarrow \mathsf{iEnc}^*(\mathsf{iMSK}_i^*, \mathsf{st}_i)\}_{i\in[\ell]}\ ,$$

where st_i contains the inner product $\mathbf{u}_{x,i}^\top \mathbf{v}_i$. We have $\mathsf{G}_0^b \approx_c \mathsf{G}_1^b$ under the SIM-security of iFE.

G_2^b **for** $b \in \{0,1\}$**:** This is the same as G_1^b except that the challenger modifies the generation of the challenge ciphertext by computing

$$\mathsf{uCT}_{x_b^*} \leftarrow \mathsf{uEnc}(\mathsf{uMPK}, x_b^*, 0^m)\ .$$

We have $\mathsf{G}_1^b \approx_c \mathsf{G}_2^b$ under the security of uABE which can be seen as follows. First of all, the adversary cannot submit an admissible combination of queries to the oracles $\mathcal{O}\mathsf{KeyGen}$ and $\mathcal{O}\mathsf{TokGen}$ that would enable a legitimate decryption of $\mathsf{uCT}_{x_b^*}$. The same holds true for $\mathcal{O}\mathsf{Update}$ queries with respect to the challenge ciphertext.

So it suffices to analyze the responses to $\mathcal{O}\mathsf{Update}$ queries for non-challenge ciphertexts. We note that such a non-challenge updated ciphertext contains iFE ciphertexts $\{\mathsf{iCT}_{x,i}\}_{i\in[\ell]}$ encrypting an update token that, if leaked, may enable a legitimate update and decryption procedure of the challenge ciphertext. Therefore, it is crucial to analyze the leakage of these iFE ciphertexts. From the property of linearly constructible updates, we know that the distribution of the iFE decryption results is the same as that of an updated ABE ciphertext. In particular, nothing else about the encrypted update tokens is leaked. Thus, under the security of ABE, we can conclude that $\mathsf{G}_1^b \approx_c \mathsf{G}_2^b$.

G_3^b **for** $b \in \{0,1\}$**:** This is the same as G_2^b except that the challenge ciphertext is computed as follows:

$$\widetilde{C} \leftarrow \mathsf{Sim}(1^\lambda, 1^{|C[\mathsf{uCT}_{x_b^*}, \{\mathsf{iSK}_{x_b^*,i}\}_{i\in[\ell]}]|}, 1^{|\mu_b^*|}) \ .$$

We have $\mathsf{G}_2^b \approx_c \mathsf{G}_3^b$ under the security of LO. Furthermore, we have $\mathsf{G}_3^0 \equiv \mathsf{G}_3^1$ since

$$|C[\mathsf{uCT}_{x_0^*}, \{\mathsf{iSK}_{x_0^*,i}\}_{i\in[\ell]}]| = |C[\mathsf{uCT}_{x_1^*}, \{\mathsf{iSK}_{x_1^*,i}\}_{i\in[\ell]}]|$$

and $|\mu_0^*| = |\mu_1^*|$.

□

5 Unbounded-Hops Ciphertext-Updatable ABE from Public-coin Evasive LWE

In this section, we present an updatable key-policy ABE scheme as well as an updatable ciphertext-policy ABE scheme allowing for an unbounded number of hops and update tokens. Both constructions rely on the hardness of public-coin evasive LWE.

5.1 Key-Policy Ciphertext-Updatable ABE

We construct a ciphertext-updatable key-policy attributed-based encryption scheme building upon a modified KP-ABE scheme in [HLL23b]. Our proposed scheme supports unbounded-hop updates and unidirectional tokens.

Construction 3 (Unbounded-hops KP-CU-ABE).

$\mathsf{Setup}(1^\lambda, 1^\ell, 1^d)$ *Given as input the security parameter* λ*, the attribute length* ℓ*, and the depth* d *of the tag-policy function family, does the following:*

1. *Sample*
$$(\mathbf{A}, \mathbf{T}_\mathbf{A}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q).$$
2. *Choose random matrices*
$$\mathbf{A}_{\mathsf{attr}} \xleftarrow{\$} \mathbb{Z}_q^{n\times \ell m}, \ \mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^n.$$
3. *Output*
$$\mathsf{mpk} := (\mathbf{A}, \mathbf{A}_{\mathsf{attr}}, \mathbf{u}), \mathsf{msk} := \mathbf{T}_\mathbf{A}.$$

$\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}, \mu \in \{0,1\})$ *Given as input the master public key, an attribute* $\mathbf{x} \in \{0,1\}^\ell$ *and a message* μ*, does the following:*

1. *Sample*
$$\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n, \ \mathbf{e}_0 \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma_0}, \ e_1 \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}, \ \mathbf{e}_{\mathsf{attr}} \leftarrow \mathcal{D}_{\mathbb{Z}^{\ell m},\sigma}.$$
2. *Compute*
$$\beta_0 = \mathbf{A}^\top \mathbf{s} + \mathbf{e}_0, \ \beta_1 = \mathbf{u}^\top \mathbf{s} + e_1 + \lceil q/2 \rfloor \mu,$$
$$\mathbf{c}_{\mathsf{attr}} = (\mathbf{A}_{\mathsf{attr}} + \mathbf{x}^\top \otimes \mathbf{G})^\top \mathbf{s} + \mathbf{e}_{\mathsf{attr}}.$$
3. *Output* $\mathsf{ct}_\mathbf{x} := (\beta_0, \beta_1, \mathbf{c}_{\mathsf{attr}})$.

$\mathsf{KeyGen}(\mathsf{msk}, f)$ *Given as input the master secret key* msk *and tag-policy* f*, does the following:*

1. *Compute* $\mathbf{H}_f := \mathsf{EvalF}(\mathbf{A}_{\mathsf{attr}}, f)$, $\mathbf{A}_f := \mathbf{A}_{\mathsf{attr}} \cdot \mathbf{H}_f$.
2. *Sample* $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_q^n$ *and generate*
$$\mathbf{k}_f := \mathbf{A}^{-1}(\mathbf{A}_f \cdot \mathbf{G}^{-1}(\mathbf{r}) + \mathbf{u})$$

using $\mathbf{T_A}$ *with standard deviation* σ_{-1}.

3. *Output* $\mathsf{sk}_f := (\mathbf{k}_f, \mathbf{r})$.

$\mathsf{TokGen}(\mathsf{msk}, \mathbf{x}, \mathbf{x}')$ *Given as input the master secret key msk, tag-attributes* $\mathbf{x}$ *and* $\mathbf{x}'$*, does the following:*

1. *Sample*
$$\mathbf{r}_{\mathbf{x}} \leftarrow \mathsf{SampleLeft}(\mathbf{A}, \mathbf{T}_{\mathbf{A}}, (\mathbf{A}_{\mathsf{attr}} + \mathbf{x}^\top \otimes \mathbf{G}), \mathbf{u}, \rho),$$
s.t. $[\mathbf{A}|\mathbf{A}_{\mathsf{attr}} + \mathbf{x}^\top \otimes \mathbf{G}] \cdot \mathbf{r}_{\mathbf{x}} = \mathbf{u}$.
2. *Sample*
$$\mathbf{S} \xleftarrow{\$} \mathbb{Z}_q^{n \times m'},\ \mathbf{E}_0 \leftarrow \mathcal{D}_{\mathbb{Z}^{m \times m'}, \sigma_0},\ \mathbf{e}_t \leftarrow \mathcal{D}_{\mathbb{Z}^{m'}, \sigma},\ \mathbf{E}_{\mathbf{x}'} \leftarrow \mathcal{D}_{\mathbb{Z}^{\ell m \times m'}, \sigma},$$
where $m' = (\ell + 1)m \log q$.
3. *Compute*
$$\mathbf{D}_0 = \mathbf{A}^\top \mathbf{S} + \mathbf{E}_0,\ \mathbf{D}_1 = \mathbf{u}^\top \mathbf{S} + \mathbf{e}_t^\top - \mathbf{r}_{\mathbf{x}}^\top \cdot \mathbf{G},$$
$$\mathbf{D}_{\mathsf{attr}} = (\mathbf{A}_{\mathsf{attr}} + \mathbf{x}'^\top \otimes \mathbf{G})^\top \mathbf{S} + \mathbf{E}_{\mathbf{x}'},$$
4. *Return* $\Delta_{\mathbf{x} \to \mathbf{x}'} = (\mathbf{D}_0, \mathbf{D}_1, \mathbf{D}_{\mathsf{attr}})$.

$\mathsf{Update}(\mathsf{ct}_{\mathbf{x}}, \Delta_{\mathbf{x} \to \mathbf{x}'})$ *Given as input a ciphertext* $\mathsf{ct}_{\mathbf{x}}$ *and an update token* $\Delta_{\mathbf{x} \to \mathbf{x}'}$*, does the following:*

1. *Parse* $\mathsf{ct}_{\mathbf{x}} := (\beta_0, \beta_1, \mathbf{c}_{\mathsf{attr}})$*, denote* $\mathbf{c}_{\mathbf{x}}$ *as* $\begin{pmatrix} \beta_0 \\ \mathbf{c}_{\mathsf{attr}} \end{pmatrix}$.
2. *Compute*
$$\beta_0' = \mathbf{D}_0 \mathbf{G}^{-1}(\mathbf{c}_{\mathbf{x}}),\ \beta_1' = \mathbf{D}_1 \mathbf{G}^{-1}(\mathbf{c}_{\mathbf{x}}) + \beta_1,$$
$$\mathbf{c}_{\mathsf{attr}}' = \mathbf{D}_{\mathsf{attr}} \mathbf{G}^{-1}(\mathbf{c}_{\mathbf{x}}).$$
3. *Output* $\mathsf{uct}_{\mathbf{x}'} = (\beta_0', \beta_1', \mathbf{c}_{\mathsf{attr}}')$.

$\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}_{\mathbf{x}}$ **or** $\mathsf{uct}_{\mathbf{x}})$ *On input a secret key* sk_f *and a ciphertext* $\mathsf{ct}_{\mathbf{x}}$ *or* $\mathsf{uct}_{\mathbf{x}}$

1. *Compute* $\mathbf{H}_{f,\mathbf{x}} := \mathsf{EvalFX}(\{\mathbf{A}_{\mathsf{attr}}\}, f, \mathbf{x})$.
2. *Compute and output*
$$\mathsf{Round}(\beta_1 - \beta_0^\top \mathbf{k}_f + \mathbf{c}_{\mathsf{attr}}^\top \cdot \mathbf{H}_{f,\mathbf{x}} \cdot \mathbf{G}^{-1}(\mathbf{r})).$$

Parameters Setting. Our parameter setting must satisfy the following constraints to guarantee the correctness and security.

1. The final magnitude of error obtained must be below $q/4$ for the correctness.
2. We must choose m large enough for running the $\mathsf{TrapGen}$ algorithm.
3. We must choose σ_{-1} and σ_0 to meet the requirements of the public-coin evasive LWE hardness assumption.
4. We must choose σ such that the LWE assumption holds.
5. We must set ρ for the $\mathsf{SampleLeft}$ algorithm.
6. We must set σ_0 to support noise flooding in the security proof.

Our parameters could be chosen as: $n = \mathsf{poly}(1^\lambda)$, $m = O(n \log q)$, $\sigma_{-1} = O(\sqrt{m \log q})$, $\rho = \omega(\sqrt{m \log m})$, $\sigma = O(\sqrt{n})$, $\sigma_0 = (\sigma + \sqrt{\ell}\sigma m^{O(d)}) \cdot \lambda^{\omega(1)}$, $B = O(m'\sigma + Q\rho m \cdot \sqrt{(\ell+1)m'} \cdot (\sigma_0 + \sigma \cdot \sqrt{\ell}))\lambda^{\omega(1)}$, $\gamma = B \cdot \lambda^{\omega(1)} + m \cdot m' \cdot \sigma_0 \cdot \sigma_{-1} \lambda^{\omega(1)} + \sigma m' \cdot m^{O(d)+1} \sqrt{\ell} \cdot \lambda^{\omega(1)}$, $q = \gamma \cdot \lambda^{\omega(1)}$.

Correctness.

Lemma 17. *For the parameters as described above, Construction 3 is correct.*

Proof. We analyze the correctness of both original and updated ciphertexts.

- For a fresh-generated ciphertext, we have

$$\begin{aligned}
&\beta_1 - \beta_0^\top \mathbf{k}_f + \mathbf{c}_{\mathsf{attr}}^\top \cdot \mathbf{H}_{f,\mathbf{x}} \cdot \mathbf{G}^{-1}(\mathbf{r}) \\
&\approx \mathbf{s}^\top \mathbf{u} + \lceil q/2 \rceil \mu - \mathbf{s}^\top (\mathbf{A}_f \cdot \mathbf{G}^{-1}(\mathbf{r}) + \mathbf{u}) + \mathbf{s}^\top (\mathbf{A}_{\mathsf{attr}} + \mathbf{x}^\top \otimes \mathbf{G}) \cdot \mathbf{H}_{f,\mathbf{x}} \mathbf{G}^{-1}(\mathbf{r}) \\
&= \lceil q/2 \rceil \mu - \mathbf{s}^\top (\mathbf{A}_f \cdot \mathbf{G}^{-1}(\mathbf{r})) + \mathbf{s}^\top \mathbf{A}_f \cdot \mathbf{G}^{-1}(\mathbf{r})
\end{aligned}$$

  The overall decryption error term is as follows:

$$\mathbf{e}_{\mathsf{dec}} = e_1 + \mathbf{e}_0^\top \cdot \mathbf{k}_f + \mathbf{e}_{\mathsf{attr}}^\top \cdot \mathbf{H}_{f,\mathbf{x}} \mathbf{G}^{-1}(\mathbf{r}).$$

  The norm of error is bounded by

$$\begin{aligned}
&|e_1| + m \cdot \|\mathbf{e}_0\| \cdot \|\mathbf{k}_f\| + \|\mathbf{e}_{\mathsf{attr}}\| \cdot \|\mathbf{H}_{f,\mathbf{x}}\| \cdot \left\|\mathbf{G}^{-1}(\mathbf{r})\right\| \\
&\leq \sigma \cdot \lambda^{\omega(1)} + m \cdot \sigma_0 \cdot \sigma_{-1} \lambda^{\omega(1)} + \sqrt{\ell m}\sigma \cdot m^{O(d)} \cdot \sqrt{m} \lambda^{\omega(1)} < \gamma.
\end{aligned}$$

  Thus, correctness holds with overwhelming probability according to our parameter setting.
- For an updated ciphertext, we rewrite the resulting uct given $\mathsf{ct}_{\mathbf{x}}$ and $\Delta_{\mathbf{x} \to \mathbf{x}'}$ as follows:

$$\begin{aligned}
\beta_0' &= (\mathbf{A}^\top \mathbf{S} + \mathbf{E}_0)\mathbf{G}^{-1}(\mathbf{c}_{\mathbf{x}}) \\
&= \mathbf{A}^\top \underbrace{\mathbf{S}\mathbf{G}^{-1}(\mathbf{c}_{\mathbf{x}})}_{\mathbf{s}'} + \underbrace{\mathbf{E}_0 \mathbf{G}^{-1}(\mathbf{c}_{\mathbf{x}})}_{\mathbf{e}_0'} \\
\beta_1' &= (\mathbf{u}^\top \mathbf{S} + \mathbf{e}_t^\top - \mathbf{r}_{\mathbf{x}}^\top \cdot \mathbf{G})\mathbf{G}^{-1}(\mathbf{c}_{\mathbf{x}}) + \beta_1 \\
&= \mathbf{u}^\top \underbrace{\mathbf{S}\mathbf{G}^{-1}(\mathbf{c}_{\mathbf{x}})}_{\mathbf{s}'} + \mathbf{e}_t^\top \mathbf{G}^{-1}(\mathbf{c}_{\mathbf{x}}) - \mathbf{r}_{\mathbf{x}}^\top \cdot \mathbf{c}_{\mathbf{x}} + \mathbf{u}^\top \mathbf{s} + e_1 + \lceil q/2 \rceil \mu \\
&= \mathbf{u}^\top \mathbf{s}' + \underbrace{\mathbf{e}_t^\top \mathbf{G}^{-1}(\mathbf{c}_{\mathbf{x}}) - \mathbf{r}_{\mathbf{x}}^\top \begin{pmatrix} \mathbf{e}_0 \\ \mathbf{e}_{\mathsf{attr}} \end{pmatrix} + e_1}_{e_1'} + \lceil q/2 \rceil \mu \\
\beta_{\mathsf{attr}}' &= ((\mathbf{A}_{\mathsf{attr}} + \mathbf{x}' \otimes \mathbf{G})^\top \mathbf{S} + \mathbf{E}_{\mathbf{x}'})\mathbf{G}^{-1}(\mathbf{c}_{\mathbf{x}}) \\
&= (\mathbf{A}_{\mathsf{attr}} + \mathbf{x}' \otimes \mathbf{G})^\top \underbrace{\mathbf{S}\mathbf{G}^{-1}(\mathbf{c}_{\mathbf{x}})}_{\mathbf{s}'} + \underbrace{\mathbf{E}_{\mathbf{x}'} \mathbf{G}^{-1}(\mathbf{c}_{\mathbf{x}})}_{\mathbf{e}_{\mathsf{attr}}'}
\end{aligned}$$

  When dealing with an updated ciphertext, similar to working with a freshly generated ciphertext ct, we also obtain a noisy result of $\lceil q/2 \rceil \mu$. However, the decryption noise is as follows:

$$\mathbf{e}_{\mathsf{dec}}' = e_1' + \mathbf{e}_0'^\top \cdot \mathbf{k}_f + \mathbf{e}_{\mathsf{attr}}'^\top \cdot \mathbf{H}_{f,\mathbf{x}} \mathbf{G}^{-1}(\mathbf{r}).$$

  The bound of the noise term in the updated ciphertexts is influenced by the number of updates. Next, we use superscripts to denote the number of updates.
  - $\mathbf{e}_0^{(0)} \sim \mathcal{D}_{\mathbb{Z}^m, \sigma_0}$, $\mathbf{e}_0^{(i)} \sim \mathcal{D}_{\mathbb{Z}^m, \sigma_0 \sqrt{m'/2}}$ for $i > 0$
  - $\mathbf{e}_{\mathsf{attr}}^{(0)} \sim \mathcal{D}_{\mathbb{Z}^{\ell m}, \sigma}$, $\mathbf{e}_{\mathsf{attr}}^{(i)} \sim \mathcal{D}_{\mathbb{Z}^{\ell m}, \sigma \sqrt{m'/2}}$ for $i > 0$
  - $e_1^{(0)} \sim \mathcal{D}_{\mathbb{Z}^m, \sigma}$
    - $e_1^{(1)} = e_1^{(0)} + \underbrace{\mathbf{e}_t^\top \mathbf{G}^{-1}(\mathbf{c}_{\mathbf{x}}^{(0)}) - \mathbf{r}_{\mathbf{x}}^\top \begin{pmatrix} \mathbf{e}_0^{(0)} \\ \mathbf{e}_{\mathsf{attr}}^{(0)} \end{pmatrix}}_{e_{\mathsf{inc}}^{(0)}}$
    - $e_1^{(i)} = e_1^{(i-1)} + \underbrace{\mathbf{e}_t^\top \mathbf{G}^{-1}(\mathbf{c}_{\mathbf{x}}^{(i-1)}) - \mathbf{r}_{\mathbf{x}}^\top \begin{pmatrix} \mathbf{e}_0^{(i-1)} \\ \mathbf{e}_{\mathsf{attr}}^{(i-1)} \end{pmatrix}}_{e_{\mathsf{inc}}}$ for $i > 1$

The error $e_1^{(Q)}$ in uct is bounded by

$$\begin{aligned}
&|e_1^{(1)}| + (Q-1)|e_{\mathsf{inc}}| \\
&= |e_1^{(0)}| + |e_{\mathsf{inc}}^{(0)}| + (Q-1)|e_{\mathsf{inc}}| \\
&= \sigma\lambda^{\omega(1)} + m' \cdot \sigma\lambda^{\omega(1)} + \sqrt{(\ell+1)m} \cdot \rho\sqrt{m \cdot \sigma_0^2 + \ell m \cdot \sigma^2}\lambda^{\omega(1)} + (Q-1)\cdot \\
&\quad \left(m' \cdot \sigma + \sqrt{(\ell+1)m} \cdot \rho\sqrt{m \cdot \left(\frac{m'}{2}\sigma_0^2\right) + \ell m \cdot \left(\frac{m'}{2}\sigma^2\right)}\right)\lambda^{\omega(1)} \\
&\leq O(m'\sigma + Q\rho m \cdot \sqrt{(\ell+1)m'} \cdot (\sigma_0 + \sigma \cdot \sqrt{\ell}))\lambda^{\omega(1)},
\end{aligned}$$

where Q is the maximum number of updates.
When $|e_1^{(Q)}| \leq B$, the total decryption error is bounded by

$$\begin{aligned}
&|e_1^{(Q)}| + m \cdot \left\|\mathbf{e}_0^{(Q)}\right\| \cdot \|\mathbf{k}_f\| + \left\|\mathbf{e}_{\mathsf{attr}}^{(Q)}\right\| \cdot \|\mathbf{H}_{f,\mathbf{x}}\| \cdot \left\|\mathbf{G}^{-1}(\mathbf{r})\right\| \\
&\leq B \cdot \lambda^{\omega(1)} + m \cdot m' \cdot \sigma_0 \cdot \sigma_{-1}\lambda^{\omega(1)} + \sqrt{\ell m} \cdot m' \cdot \sigma \cdot m^{O(d)} \cdot \sqrt{m}\lambda^{\omega(1)} \leq \gamma.
\end{aligned}$$

Therefore, correctness holds with our parameters setting.

□

Security.

Theorem 4. *Assuming the hardness of public-coin evasive LWE (Definition 3) and LWE (Definition 1), the proposed updatable KP-ABE scheme in Construction 3 is very selectively secure.*

Proof. Suppose that the adversary $\mathcal{A}$ with randomness $\mathsf{coin}_{\mathcal{A}}$ submits the challenge attribute $\mathbf{x}^*$, secret key queries $\mathcal{K} = \{f_k\}_{k\in[Q_{\mathsf{sk}}]}$, update token queries $\mathcal{CT} = \{\mathbf{x}_{i,\text{before}} \to \mathbf{x}_{i,\text{after}}\}_{i\in[Q_{\mathsf{ct}}]}$ and updating ciphertext queries $\mathcal{D} = \{\mathbf{x}_{j,\text{before}} \to \mathbf{x}_{j,\text{after}}\}_{j\in[Q_{\mathsf{uct}}]}$ at the very beginning.

From the update token queries, we define a subset $\mathcal{CT}^* = \{\mathbf{x}^*_{i,\text{before}} \to \mathbf{x}^*_{i,\text{after}}\}_{i\in[Q^*_{\mathsf{ct}}]}$ containing all attributes that appear in the update chain starting from the challenge attribute $\mathbf{x}^*$. Similarly, from the updating ciphertext queries, we derive a subset $\mathcal{D}^* = \{\mathbf{x}^* \to \mathbf{x}^*_j\}_{j\in[Q^*_{\mathsf{uct}}]}$, which consists of all updates where the before-update ciphertext corresponds to the challenge ciphertext ct^*. Define a set $\mathcal{UC} := \mathbf{x}^* \cup \mathcal{D}^* \cup \mathcal{CT}^*$ and denote $|\mathcal{UC}|$ by Q_{uc}. Note that only the update tokens in $\mathcal{CT}^*$ and the updated ciphertexts in $\mathcal{D}^*$ could potentially help the adversary in learning information about the challenge bit.

In order to rely on $\mathsf{PublicEvLWE}$ assumption, we consider the pre-experiment $\mathsf{pre}^b_{\mathcal{B}}(\lambda)$ and post-experiment $\mathsf{post}^b_{\mathcal{A}}(\lambda)$, as showed in Figure 6 and Figure 7. In both experiments, the grayboxed components are chosen from corresponding uniformly random distribution. The security of CU-KP-ABE scheme explicitly follows from the indistinguishability of $\mathsf{post}^b_{\mathcal{A}}(\lambda)$, which implies that the challenge ciphertext, update tokens in $\mathcal{CT}^*$ and updated ciphertexts in $\mathcal{D}^*$ are distributed close to uniformly random. By relying on the public-coin evasive LWE assumption, it suffices to prove the pseudorandomness of the output of $\mathsf{pre}^b_{\mathcal{B}}(\lambda)$.

The proof proceeds via a sequence of hybrids $\mathsf{G}^b_0, \mathsf{G}^b_1, \mathsf{G}^b_2, \mathsf{G}^b_3$ for $b \in \{0,1\}$.

G^b_0 **for** $b \in \{0,1\}$**:** This is the original experiment for challenge bit b.
G^b_1 **for** $b \in \{0,1\}$**:** The challenge ciphertext components are switched to random. To show that $\mathsf{G}^b_0 \approx_c \mathsf{G}^b_1$, we rewrite the view of the adversary $\mathcal{A}$ as

$$\mathsf{mpk} = (\mathbf{A}, \mathbf{A}_{\mathsf{attr}}, \mathbf{u}),\ \beta'^*_1 = \mathbf{u}^\top \mathbf{s}_{\hat{\imath}^*} + e_{1,\hat{\imath}^*}$$
$$\mathbf{c}^*_{\mathsf{attr}} = (\mathbf{A}_{\mathsf{attr}} + \mathbf{x}^{*\top} \otimes \mathbf{G})^\top \mathbf{s}_{\hat{\imath}^*} + \mathbf{e}_{\mathsf{attr},\hat{\imath}^*},\ \{\beta^*_{0,k} = (\mathbf{A}_{f,k}\mathbf{G}^{-1}(\mathbf{r}_k) + \mathbf{u})^\top \mathbf{s}_{\hat{\imath}^*} + \mathbf{e}'_{0,\hat{\imath}^*,k}\}_{k\in[Q_{\mathsf{sk}}]}.$$

Define the following intermediate hybrid games.

– $\mathsf{G}^b_{1,0}$: Same as G^b_0.

$\mathsf{pre}^b_{\mathcal{B}}(\lambda)$

$\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$

$(\{\mathbf{A}'_{\mathsf{attr},\hat{i}}\}_{\hat{i}\in Q_{\mathsf{uc}}}, \mathbf{u}, \{\mathbf{p}_k\}_{k\in[Q_{\mathsf{sk}}]}, \mathsf{aux}) \leftarrow \mathsf{Samp}(1^\lambda; \mathsf{rand})$

$\mathbf{A}'_{\mathsf{attr},\hat{i}} = \mathbf{A}_{\mathsf{attr}} - \mathbf{x}_{\hat{i}}^\top \otimes \mathbf{G}$ for all $\mathbf{x}_{\hat{i}} \in \mathcal{UC}$ (denote $\mathbf{x}^*$ by $\mathbf{x}_{\hat{i}^*}$)

$\mathbf{H}_{f,k} \leftarrow \mathsf{EvalF}(\mathbf{A}_{\mathsf{attr}}, f_k), \mathbf{A}_{f,k} \coloneqq \mathbf{A}_{\mathsf{attr}} \cdot \mathbf{H}_{f,k}, \mathbf{p}_k = \mathbf{A}_{f,k}\mathbf{G}^{-1}(\mathbf{r}_k) + \mathbf{u}$ for all $f_k \in \mathcal{K}$

$\mathsf{aux} = (\mathbf{A}_{\mathsf{attr}}, \{\mathbf{A}_{f,k}, \mathbf{r}_k\}_{k\in[Q_{\mathsf{sk}}]}, \mathsf{coin}_{\mathcal{A}}, \mathbf{x}^*, \{\mathbf{x}^*_j\}_{j\in[Q^*_{\mathsf{uct}}]}, \{\mathbf{x}^*_{i,\mathrm{after}}\}_{i\in[Q^*_{\mathsf{ct}}]}, \{f_k\}_{k\in[Q_{\mathsf{sk}}]})$

$\mathbf{s}_{\hat{i}^*} \xleftarrow{\$} \mathbb{Z}_q^n, \mathbf{S}_{\hat{i}} \xleftarrow{\$} \mathbb{Z}_q^{n\times m'}$

if $b = 0$ **then**

$\mathbf{e}_{0,\hat{i}^*} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma_0}, e'_{0,\hat{i}^*,k} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma_0}, \mathbf{e}_{\mathsf{attr},\hat{i}^*} \leftarrow \mathcal{D}_{\mathbb{Z}^{\ell m},\sigma}, e_{1,\hat{i}^*} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}$,

$\beta^*_0 \coloneqq \mathbf{A}^\top \mathbf{s}_{\hat{i}^*} + \mathbf{e}_{0,\hat{i}^*}, \{\beta^*_{0,k} \coloneqq \mathbf{p}_k^\top \mathbf{s}_{\hat{i}^*} + e'_{0,\hat{i}^*,k}\}_{k\in Q_{\mathsf{sk}}}, \mathbf{c}^*_{\mathsf{attr}} \coloneqq \mathbf{A}'^\top_{\mathsf{attr},\hat{i}^*}\mathbf{s}_{\hat{i}^*} + \mathbf{e}_{\mathsf{attr},\hat{i}^*}$,

$\beta'^*_1 \coloneqq \mathbf{u}^\top \mathbf{s}_{\hat{i}^*} + e_{1,\hat{i}^*}$

$\{\mathbf{E}_{0,\hat{i}} \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m'},\sigma_0}, \mathbf{e}'_{0,\hat{i},k} \leftarrow \mathcal{D}_{\mathbb{Z}^{m'},\sigma_0}, \mathbf{E}_{\mathsf{attr},\hat{i}} \leftarrow \mathcal{D}_{\mathbb{Z}^{\ell m\times m'},\sigma}, \mathbf{e}_{1,\hat{i}} \leftarrow \mathcal{D}_{\mathbb{Z}^{m'},\sigma}\}_{\hat{i}\in[Q_{\mathsf{uc}}]\setminus\hat{i}^*}$

$$\left\{\begin{array}{l}\mathbf{D}_{0,\hat{i}} \coloneqq \mathbf{A}^\top\mathbf{S}_{\hat{i}} + \mathbf{E}_{0,\hat{i}}, \widetilde{\mathbf{D}}_{k,\hat{i}} \coloneqq \mathbf{p}_k^\top\mathbf{S}_{\hat{i}} + \mathbf{e}'_{0,\hat{i},k},\\ \mathbf{D}'_{\mathsf{attr},\hat{i}} \coloneqq \mathbf{A}'^\top_{\mathsf{attr},\hat{i}}\mathbf{S}_{\hat{i}} + \mathbf{E}_{\mathsf{attr},\hat{i}}, \mathbf{D}'_{1,\hat{i}} \coloneqq \mathbf{u}^\top\mathbf{S}_{\hat{i}} + \mathbf{e}_{1,\hat{i}}\end{array}\right\}_{\hat{i}\in[Q_{\mathsf{uc}}]\setminus\hat{i}^*,k\in[Q_{\mathsf{sk}}]}$$

if $b = 1$ **then**

$$\beta^*_0,\ \{\beta^*_{0,k}\}_{k\in Q_{\mathsf{sk}}},\ \mathbf{c}^*_{\mathsf{attr}},\ \beta'^*_1,\ \left\{\mathbf{D}_{0,\hat{i}},\ \widetilde{\mathbf{D}}_{k,\hat{i}},\ \mathbf{D}'_{\mathsf{attr},\hat{i}},\ \mathbf{D}'_{1,\hat{i}}\right\}_{\hat{i}\in[Q_{\mathsf{uc}}]\setminus\hat{i}^*,k\in[Q_{\mathsf{sk}}]}$$

return
$$\left(\begin{array}{c}\mathbf{A},\quad \{\mathbf{A}'_{\mathsf{attr},\hat{i}}\}_{\hat{i}\in Q_{\mathsf{uc}}},\quad \mathbf{u},\\ \beta'^*_1, \{\mathbf{D}_{0,\hat{i}}\}_{\hat{i}\in[Q_{\mathsf{uc}}]\setminus\hat{i}^*}, \mathbf{c}'^*_{\mathsf{attr}}, \{\mathbf{D}'_{\mathsf{attr},\hat{i}}\}_{\hat{i}\in[Q_{\mathsf{uc}}]\setminus\hat{i}^*}, \beta'^*_1, \{\mathbf{D}'_{1,\hat{i}}\}_{\hat{i}\in[Q_{\mathsf{uc}}]\setminus\hat{i}^*},\\ \{\mathbf{p}_k\}_{k\in[Q_{\mathsf{sk}}]},\\ \{\beta^*_{0,k}, \widetilde{\mathbf{D}}_{k,\hat{i}}\}_{\hat{i}\in[Q_{\mathsf{uc}}]\setminus\hat{i}^*,k\in[Q_{\mathsf{sk}}]}, \mathsf{aux}\end{array}\right)$$

(In the above experiment, $\mathbf{A}, \mathbf{A}_{\mathsf{attr}}, \mathbf{u}, \{\mathbf{r}_k\}_{k\in[Q_{\mathsf{sk}}]}$ are sampled as in normal scheme.)

Fig. 6. Experiment $\mathsf{pre}^b_{\mathcal{B}}(\lambda)$ for PublicEvLWE.

– $\mathsf{G}^b_{1,1}$: Same as $\mathsf{G}^b_{1,0}$, except we compute

$$\beta^*_{0,k} = \beta'^*_1 + \mathbf{c}^{*\top}_{\mathsf{attr}} \cdot \mathbf{H}_{f_k,\mathbf{x}^*} \cdot \mathbf{G}^{-1}(\mathbf{r}_k) - \mathbf{s}^\top_{\hat{i}^*}\mathbf{r}_{\mathbf{k}} + e'_{0,\hat{i}^*,k}.$$

We observe that

$$\begin{aligned}&\mathbf{s}^\top_{\hat{i}^*}(\mathbf{A}_{f,k}\mathbf{G}^{-1}(\mathbf{r}_k) + \mathbf{u})\\ =\ &\underbrace{\mathbf{s}^\top_{\hat{i}^*}(\mathbf{A}_{\mathsf{attr}} + \mathbf{x}^{*\top}\otimes\mathbf{G})}_{\approx \mathbf{c}^{*\top}_{\mathsf{attr}}}\mathbf{H}_{f_k,\mathbf{x}^*}\mathbf{G}^{-1}(\mathbf{r}_k) - \mathbf{s}^\top_{\hat{i}^*} \cdot f_k(\mathbf{x}^*)\mathbf{G}\cdot\mathbf{G}^{-1}(\mathbf{r}_k) + \underbrace{\mathbf{s}^\top_{\hat{i}^*}\mathbf{u}}_{\approx \beta'^*_1}.\end{aligned}$$

Using $e'_{0,\hat{i}^*,k}$ for noise flooding, we have

$$e'_{0,\hat{i}^*,k} \approx_s e'_{0,\hat{i}^*,k} + e_{1,\hat{i}^*} + e^\top_{\mathsf{attr},\hat{i}^*} \cdot \mathbf{H}_{f_k,\mathbf{x}^*}\mathbf{G}^{-1}(\mathbf{r}_k),$$

where $\mathbf{e}'_{0,\hat{i}^*,k} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma_0}$ and $\sigma_0 > (\sigma + \sqrt{\ell}\sigma m^{O(d)})\cdot\lambda^{\omega(1)}$. Hence, $\mathsf{G}^b_{1,1} \approx_s \mathsf{G}^b_{1,0}$.

– $\mathsf{G}^b_{1,2}(\equiv \mathsf{G}^b_1)$: Same as $\mathsf{G}^b_{1,1}$, except that we sample $\beta'^*_1, \mathbf{c}^*_{\mathsf{attr}}, \beta^*_{0,k}$ from corresponding uniformly random distributions. We first program $\mathbf{A}_{\mathsf{attr}}$ as $\mathbf{A}_{\mathsf{attr}} - \mathbf{x}^{*\top}\otimes\mathbf{G}$ relying on the fact that $\mathbf{A}_{\mathsf{attr}}$ is sampled

$\mathsf{post}^b_{\mathcal{A}}(\lambda)$

$\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$

$(\{\mathbf{A}'_{\mathsf{attr},\hat{\imath}}\}_{\hat{\imath}\in Q_{\mathsf{uc}}}, \mathbf{u}, \{\mathbf{p}_k\}_{k\in[Q_{\mathsf{sk}}]}, \mathsf{aux}) \leftarrow \mathsf{Samp}(1^\lambda; \mathsf{rand})$ as in $\mathsf{pre}^b_{\mathcal{B}}(\lambda)$

$\mathbf{s}_{\hat{\imath}^*} \xleftarrow{\$} \mathbb{Z}_q^n, \mathbf{S}_{\hat{\imath}} \xleftarrow{\$} \mathbb{Z}_q^{n\times m'}$

if $b = 0$ **then**

Sample $\mathbf{e}_{0,\hat{\imath}^*}, \mathbf{e}_{\mathsf{attr},\hat{\imath}^*}, e_{1,\hat{\imath}^*}$ as in $\mathsf{pre}^b_{\mathcal{B}}(\lambda)$,

$\beta^*_0 := \mathbf{A}^\top \mathbf{s}_{\hat{\imath}^*} + \mathbf{e}_{0,\hat{\imath}^*}, \mathbf{c}^*_{\mathsf{attr}} := \mathbf{A}'^\top_{\mathsf{attr},\hat{\imath}^*}\mathbf{s}_{\hat{\imath}^*} + \mathbf{e}_{\mathsf{attr},\hat{\imath}^*}, \beta'^*_1 := \mathbf{u}^\top \mathbf{s}_{\hat{\imath}^*} + e_{1,\hat{\imath}^*},$

Sample $\{\mathbf{E}_{0,\hat{\imath}}, \mathbf{E}_{\mathsf{attr},\hat{\imath}}, \mathbf{e}_{1,\hat{\imath}} \leftarrow \mathcal{D}_{\mathbb{Z}^{m'},\sigma}\}_{\hat{\imath}\in[Q_{\mathsf{uc}}]\setminus\hat{\imath}^*}$ as in $\mathsf{pre}^b_{\mathcal{B}}(\lambda)$,

$$\left\{\begin{array}{c}\mathbf{D}_{0,\hat{\imath}} := \mathbf{A}^\top\mathbf{S}_{\hat{\imath}} + \mathbf{E}_{0,\hat{\imath}}, \mathbf{D}'_{\mathsf{attr},\hat{\imath}} := \mathbf{A}'^\top_{\mathsf{attr},\hat{\imath}}\mathbf{S}_{\hat{\imath}} + \mathbf{E}_{\mathsf{attr},\hat{\imath}},\\ \mathbf{D}'_{1,\hat{\imath}} := \mathbf{u}^\top\mathbf{S}_{\hat{\imath}} + \mathbf{e}_{1,\hat{\imath}}\end{array}\right\}_{\hat{\imath}\in[Q_{\mathsf{uc}}]\setminus\hat{\imath}^*]}$$

$\{\mathbf{A}^{-1}(\mathbf{A}_{f_k}\cdot\mathbf{G}^{-1}(\mathbf{r}_k) + \mathbf{u})\}_{k\in Q_{\mathsf{sk}}}$

if $b = 1$ **then**

$$\beta^*_0,\ \mathbf{c}^*_{\mathsf{attr}},\ \beta'^*_1,\ \left\{\mathbf{D}_{0,\hat{\imath}},\ \mathbf{D}'_{\mathsf{attr},\hat{\imath}},\ \mathbf{D}'_{1,\hat{\imath}}\right\}_{\hat{\imath}\in[Q_{\mathsf{uc}}]\setminus\hat{\imath}^*]}$$

$\{\mathbf{A}^{-1}(\mathbf{A}_{f_k}\cdot\mathbf{G}^{-1}(\mathbf{r}_k) + \mathbf{u})\}_{k\in Q_{\mathsf{sk}}}$

return
$$\left(\begin{array}{c}\mathbf{A},\quad \{\mathbf{A}'_{\mathsf{attr},\hat{\imath}}\}_{\hat{\imath}\in Q_{\mathsf{uc}}},\quad \mathbf{u},\\ \beta'^*_1, \{\mathbf{D}_{0,\hat{\imath}}\}_{\hat{\imath}\in[Q_{\mathsf{uc}}]\setminus\hat{\imath}^*}, \mathbf{c}'^*_{\mathsf{attr}}, \{\mathbf{D}'_{\mathsf{attr},\hat{\imath}}\}_{\hat{\imath}\in[Q_{\mathsf{uc}}]\setminus\hat{\imath}^*}, \beta'^*_1, \{\mathbf{D}'_{1,\hat{\imath}}\}_{\hat{\imath}\in[Q_{\mathsf{uc}}]\setminus\hat{\imath}^*},\\ \{\mathbf{p}_k\}_{k\in[Q_{\mathsf{sk}}]},\\ \{\mathbf{A}^{-1}(\mathbf{A}_{f_k}\cdot\mathbf{G}^{-1}(\mathbf{r}_k) + \mathbf{u})\}_{k\in Q_{\mathsf{sk}}}, \mathsf{aux}\end{array}\right)$$

(In the above experiment, $\mathbf{A}, \mathbf{A}_{\mathsf{attr}}, \mathbf{u}, \{\mathbf{r}_k\}_{k\in[Q_{\mathsf{sk}}]}$ are sampled as in normal scheme.)

Fig. 7. Experiment $\mathsf{post}^b_{\mathcal{A}}(\lambda)$ for PublicEvLWE.

uniformly from $\mathbb{Z}_q^{n\times \ell m}$. According to the LWE assumptions [Reg05, BLMR13], we have

$$\begin{aligned}
&(\mathbf{A}_{\mathsf{attr}} - \mathbf{x}^{*\top}\otimes\mathbf{G}, \mathbf{u}, \mathbf{r}_k, \begin{pmatrix}(\mathbf{A}_{\mathsf{attr}} + \mathbf{x}^{*\top}\otimes\mathbf{G})^\top\mathbf{s}_{\hat{\imath}^*} + \mathbf{e}_{\mathsf{attr},\hat{\imath}^*},\\ \mathbf{u}^\top\mathbf{s}_{\hat{\imath}^*} + e_{1,\hat{\imath}^*},\ \mathbf{r}_k^\top\mathbf{s}_{\hat{\imath}^*} + e'_{0,\hat{\imath}^*,k}\end{pmatrix})\\
\approx_s\ &(\mathbf{A}_{\mathsf{attr}} - \mathbf{x}^{*\top}\otimes\mathbf{G}, \mathbf{u}, \mathbf{r}_k, \left(\mathbf{c}^*_{\mathsf{attr}},\ \beta'^*_1,\ \beta^*_{0,k}\right))\\
\approx_s\ &(\mathbf{A}_{\mathsf{attr}}, \mathbf{u}, \mathbf{r}_k, \left(\mathbf{c}^*_{\mathsf{attr}},\ \beta'^*_1,\ \beta^*_{0,k}\right))
\end{aligned}$$

The grayboxed components are chosen from corresponding uniformly random distribution. The last $\approx_s$ also relies on the uniform distribution of $\mathbf{A}_{\mathsf{attr}}$. Therefore, we have $\mathsf{G}^b_{1,2} \approx_c \mathsf{G}^b_{1,1}$.

G^b_2 **for** $b \in \{0,1\}$**:** The components of updated tokens for queries with index $\hat{\imath} \in Q^*_{\mathsf{uct}} \cup Q^*_{\mathsf{ct}}$ are switched to random. We define the intermediate hybrid games G^b_{2,i_u} over $i_u \in Q^*_{\mathsf{uct}} \cup Q^*_{\mathsf{ct}}$, in which the updated tokens $\{\Delta_{\hat{\imath}}\}_{\hat{\imath}\le i_u}$ are sampled from uniformly random while the remaining $\{\Delta_{\hat{\imath}}\}_{\hat{\imath}>i_u}$ are computed as in G^b_1. Thus, $\mathsf{G}^b_{2,i_u} \approx_c \mathsf{G}^b_{2,i_u-1}$ follows similarly from the analysis of $\mathsf{G}^b_0 \approx_s \mathsf{G}^b_1$. Therefore, we have $\mathsf{G}^b_2 \approx_c \mathsf{G}^b_1$.

G^b_3 **for** $b \in \{0,1\}$**:** The components of updated ciphertexts starting from the challenge ciphertext are switched to random. $\mathsf{G}^b_3 \equiv \mathsf{G}^b_2$ follows from the fact that at the end of G^b_2, all the update tokens $\Delta_{\hat{\imath}}$ and ct^* being required to compute $\mathsf{uct}_{\hat{\imath}}$ have already been replaced by uniformly random, ensuring that the resulting $\mathsf{uct}_{\hat{\imath}}$ are distributed as uniformly random. Moreover, as all the components containing b in the game transcript have been switched to random, the advantage of the adversary $\mathcal{A}$ is now zero in G^b_3. We complete the proof.

□

5.2 Ciphertext-Policy Ciphertext-Updatable ABE

We construct a ciphertext-updatable CP-ABE scheme for poly-depth circuits based on public-coin evasive LWE and tensor LWE, leveraging the CP-ABE construction from [Wee22] and our proposed ciphertext update techniques. Similarly, the proposed scheme supports unbounded-updates and unidirectional tokens. We can eliminate the dependency on the tensor LWE by replacing gadget matrices $\mathbf{G}$ in the scheme with identity matrices $\mathbf{I}_m$ and using a low-norm matrix $\mathbf{A}_{\mathsf{attr}}$, thereby obtain an updatable CP-ABE scheme for NC^1 circuits.

Construction 4 (Unbounded-hops CP-CU-ABE).

$\mathsf{Setup}(1^\lambda, 1^\ell, 1^d)$ *Given as input the security parameter λ, the attribute length ℓ, and the depth d of the tag-policy function family, does the following:*

1. *Sample*
$$(\mathbf{B}, \mathbf{T}_{\mathbf{B}}) \leftarrow \mathsf{TrapGen}(1^{nm+n}, 1^{m'}, q),$$
where $m' = O((nm+n)\log q)$.
2. *Choose random matrices*
$$\mathbf{B}_0 \xleftarrow{\$} \mathbb{Z}_q^{n\times m},\ \mathbf{A}_{\mathsf{attr}} \xleftarrow{\$} \mathbb{Z}_q^{n\times \ell m},\ \mathbf{u} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma'}.$$
3. *Output*
$$\mathsf{mpk} \coloneqq (\mathbf{B}, \mathbf{A}_{\mathsf{attr}}, \mathbf{B}_0, \mathbf{u}), \mathsf{msk} \coloneqq (\mathbf{T}_{\mathbf{A}}, \mathbf{T}_{\mathbf{B}}).$$

$\mathsf{Enc}(\mathsf{mpk}, f, \mu \in \{0,1\})$ *Given as input the master public key, a policy f and a message μ, does the following:*

1. *Compute* $\mathbf{H}_f \coloneqq \mathsf{EvalF}(\mathbf{A}_{\mathsf{attr}}, f)$, $\mathbf{A}_f \coloneqq \mathbf{A}_{\mathsf{attr}} \cdot \mathbf{H}_f$.
2. *Sample*
$$\mathbf{s}_0 \xleftarrow{\$} \mathbb{Z}_q^{nm},\ \mathbf{s}_1 \xleftarrow{\$} \mathbb{Z}_q^{n},$$
$$\mathbf{e}_0 \leftarrow \mathcal{D}_{\mathbb{Z}^{m'},\sigma},\ \mathbf{e} \leftarrow \mathcal{D}_{\mathbb{Z}^{m},\sigma}.$$
3. *Compute*
$$\beta_0 = \mathbf{B}^\top \begin{pmatrix}\mathbf{s}_0\\ \mathbf{s}_1\end{pmatrix} + \mathbf{e}_0,$$
$$\mathbf{c}_f = (\mathbf{A}_f\mathbf{u} \otimes \mathbf{I}_m)^\top \mathbf{s}_0 + \mathbf{B}_0^\top \mathbf{s}_1 + \mu \cdot \mathbf{g} + \mathbf{e}.$$
4. *Output* $\mathsf{ct}_f \coloneqq (\beta_0, \mathbf{c}_f)$.

$\mathsf{KeyGen}(\mathsf{msk}, \mathbf{x})$ *Given as input the master secret key* msk *and attribute $\mathbf{x} \in \{0,1\}^\ell$, does the following:*

1. *Sample $\mathbf{r} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma'}$ and generate*
$$\mathbf{K} \coloneqq \mathbf{B}^{-1}\begin{pmatrix}(\mathbf{A}_{\mathsf{attr}} + \mathbf{x}^\top \otimes \mathbf{G}) \otimes \mathbf{r}\\ \mathbf{B}_0\mathbf{r}\end{pmatrix}$$
using $\mathbf{T}_{\mathbf{B}}$ with standard deviation σ_{-1}.
2. *Output* $\mathsf{sk}_{\mathbf{x}} \coloneqq (\mathbf{K}, \mathbf{r})$.

$\mathsf{TokGen}(\mathsf{msk}, f, f')$ *Given as input the master secret key msk, policies f and f', does the following:*

1. *Sample*
$$\mathbf{R} \coloneqq \mathbf{B}^{-1}\begin{pmatrix}\mathbf{A}_f\mathbf{u} \otimes \mathbf{I}_m\\ \mathbf{B}_0\end{pmatrix}$$
using $\mathbf{T}_{\mathbf{B}}$ with standard deviation σ_{-1}.
2. *Sample*

$$\mathbf{S}_0 \xleftarrow{\$} \mathbb{Z}_q^{nm\times m''},\ \mathbf{S}_1 \xleftarrow{\$} \mathbb{Z}_q^{n\times m''},$$
$$\mathbf{E}_0 \leftarrow \mathcal{D}_{\mathbb{Z}^{m'\times m''},\sigma},\ \mathbf{E} \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times m''},\sigma},$$

where $m'' = O(m' \log q)$.

3. *Compute*

$$\mathbf{D}_0 = \mathbf{B}^\top \begin{pmatrix}\mathbf{S}_0\\ \mathbf{S}_1\end{pmatrix} + \mathbf{E}_0,$$
$$\mathbf{D}_{f'} = (\mathbf{A}_{f'}\mathbf{u}\otimes \mathbf{I}_m)^\top \mathbf{S}_0 + \mathbf{B}_0^\top \mathbf{S}_1 + \mathbf{E} - \mathbf{R}^\top \mathbf{G},$$

4. *Return* $\Delta_{f\to f'} = (\mathbf{D}_0, \mathbf{D}_{f'})$.

$\mathsf{Update}(\mathsf{ct}_f, \Delta_{f\to f'})$ *Given as input a ciphertext* ct_f *and an update token* $\Delta_{f\to f'}$*, does the following:*

1. *Parse* $\mathsf{ct}_f := (\beta_0, \mathbf{c}_f)$.
2. *Compute*

$$\beta_0' = \mathbf{D}_0\mathbf{G}^{-1}(\beta_0),\ \mathbf{c}_{f'}' = \mathbf{D}_{f'}\mathbf{G}^{-1}(\beta_0) + \mathbf{c}_f.$$

3. *Output* $\mathsf{uct}_{f'} = (\beta_0', \mathbf{c}_{f'}')$.

$\mathsf{Dec}(\mathsf{sk}_\mathbf{x}, \mathsf{ct}_f$ **or** $\mathsf{uct}_f)$ *On input a secret key* $\mathsf{sk}_\mathbf{x}$ *and a ciphertext* ct_f *or* uct_f

1. *Compute* $\mathbf{H}_{f,\mathbf{x}} := \mathsf{EvalFX}(\{\mathbf{A}_{\mathsf{attr}}\}, f, \mathbf{x})$.
2. *Compute and output*

$$\mathsf{Round}_\gamma(\mathbf{c}_f^\top \cdot \mathbf{r} - \beta_0^\top \mathbf{K} \cdot \begin{pmatrix}\mathbf{H}_{f,\mathbf{x}}\cdot \mathbf{u}\\ 1\end{pmatrix}),$$

where $\mathsf{Round}_\gamma(\mu')$ *outputs* 0 *if* $|\mu'| < \gamma$ *and* 1 *otherwise.*

Parameters Setting. Our parameters can be chosen analogously to [Wee22], except that we need a larger modulus q to handle multi-hop updates. We set the parameters to satisfy the following constraints aiming for the correctness and security.

1. The final magnitude of error obtained must be below γ for the correctness.
2. We must choose m' large enough for running the $\mathsf{TrapGen}$ algorithm.
3. We must choose σ such that the public-coin evasive LWE assumption holds.
4. We must choose σ_{-1} to meet the requirement of the $\mathsf{SamplePre}$ algorithm.
5. We must set σ to support noise flooding in the security proof.

Our parameters could be chosen as: $n = \mathsf{poly}(1^\lambda)$, $m = O(n\log q)$, $m' = O((nm+n)\log q)$, $\sigma = m^{O(d)}\cdot\lambda^{\omega(1)}$, $\sigma'' = \lambda^{\omega(1)}$, $\sigma' = \mathsf{poly}(\lambda)$, $\sigma_{-1} = O(\sqrt{(nm+n)\cdot\log q})$, $\gamma = B\sigma'\sqrt{m}\lambda^{\omega(1)} + \sigma\sqrt{\frac{m''}{2}}O(\sigma_{-1}\sigma')m^{O(d)+1}\lambda^{\omega(1)}$, $B = Q\cdot O\left(\sigma\sqrt{\frac{m''}{2}}m\right)\lambda^{\omega(1)}$, $q = \gamma\cdot\mathsf{poly}(\lambda)$.

Correctness.

Lemma 18. *Under the parameters set as described above, Construction 4 is correct.*

Proof. The correctness for a fresh ciphertext follows from the correctness of the CP-ABE scheme [Wee22]. Therefore, we focus on analyzing the correctness for updated ciphertexts.

For an updated ciphertext, we rewrite the resulting uct given ct_f and $\Delta_{f\to f'}$ as follows:

$$\begin{aligned}\beta_0' &= (\mathbf{B}^\top \begin{pmatrix}\mathbf{S}_0\\ \mathbf{S}_1\end{pmatrix} + \mathbf{E}_0)\mathbf{G}^{-1}(\beta_0)\\ &= \mathbf{B}^\top \begin{pmatrix}\underbrace{\mathbf{S}_0\mathbf{G}^{-1}(\beta_0)}_{\mathbf{s}_0'}\\ \underbrace{\mathbf{S}_1\mathbf{G}^{-1}(\beta_0)}_{\mathbf{s}_1'}\end{pmatrix} + \underbrace{\mathbf{E}_0\mathbf{G}^{-1}(\beta_0)}_{\mathbf{e}_0'}\end{aligned}$$

$$
\begin{aligned}
\mathbf{c}'_{f'} &= ((\mathbf{A}_{f'}\mathbf{u}\otimes\mathbf{I}_m)^\top\mathbf{S}_0 + \mathbf{B}_0^\top\mathbf{S}_1 + \mathbf{E} - \mathbf{R}^\top\mathbf{G} + \mathbf{c}_f \\
&= (\mathbf{A}_{f'}\mathbf{u}\otimes\mathbf{I}_m)^\top\mathbf{s}'_0 + \mathbf{B}_0^\top\mathbf{s}'_1 + \mathbf{E}\mathbf{G}^{-1}(\beta_0) - \mathbf{R}^\top\beta_0 + (\mathbf{A}_f\mathbf{u}\otimes\mathbf{I}_m)^\top\mathbf{s}_0 \\
&\quad + \mathbf{B}_0^\top\mathbf{s}_1 + \mu\cdot\mathbf{g} + \mathbf{e} \\
&= (\mathbf{A}_{f'}\mathbf{u}\otimes\mathbf{I}_m)^\top\mathbf{s}'_0 + \mathbf{B}_0^\top\mathbf{s}'_1 + \mu\cdot\mathbf{g} + \underbrace{\mathbf{e} + \mathbf{E}\mathbf{G}^{-1}(\beta_0) - \mathbf{R}^\top\mathbf{e}_0}_{\mathbf{e}'}
\end{aligned}
$$

When dealing with an updated ciphertext, similar to working with a freshly generated ciphertext ct, we also obtain a noisy result of $\mu\cdot\mathbf{g}^\top\mathbf{r}$. For $f', \mathbf{x}$ satisfying $f'(\mathbf{x}) = 0$, we have

$$
\begin{aligned}
&\beta_0'^\top\mathbf{K}\cdot\begin{pmatrix}\mathbf{H}_{f',\mathbf{x}}\cdot\mathbf{u}\\ 1\end{pmatrix} \\
&\approx (\mathbf{s}_0'^\top|\mathbf{s}_1'^\top)\mathbf{B}\cdot\mathbf{B}^{-1}\begin{pmatrix}(\mathbf{A}_{\mathsf{attr}}+\mathbf{x}^\top\otimes\mathbf{G})\otimes\mathbf{r} \\ \mathbf{B}_0\mathbf{r}\end{pmatrix}\cdot\begin{pmatrix}\mathbf{H}_{f',\mathbf{x}}\cdot\mathbf{u}\\ 1\end{pmatrix} \\
&= (\mathbf{s}_0'^\top((\mathbf{A}_{\mathsf{attr}}+\mathbf{x}^\top\otimes\mathbf{G})\otimes\mathbf{r}) \mid \mathbf{s}_1'^\top\mathbf{B}_0\mathbf{r})\cdot\begin{pmatrix}\mathbf{H}_{f',\mathbf{x}}\cdot\mathbf{u}\\ 1\end{pmatrix} \\
&\quad \downarrow (\mathbf{A}\otimes\mathbf{B})\otimes(\mathbf{C}\otimes\mathbf{D}) = \mathbf{AC}\otimes\mathbf{BD} \\
&= \mathbf{s}_0'^\top((\mathbf{A}_{\mathsf{attr}}+\mathbf{x}^\top\otimes\mathbf{G})\cdot\mathbf{H}_{f',\mathbf{x}}\mathbf{u}\cdot 1)\otimes(\mathbf{I}_m\cdot\mathbf{r}) + \mathbf{s}_1'^\top\mathbf{B}_0\mathbf{r} \\
&= \mathbf{s}_0'^\top(\mathbf{A}_{f'}\mathbf{u}\otimes\mathbf{I}_m)\cdot\mathbf{r} + \mathbf{s}_1'^\top\mathbf{B}_0\mathbf{r}
\end{aligned}
$$

Hence,

$$
\begin{aligned}
&\mathbf{c}_f'^\top\cdot\mathbf{r} - \beta_0'^\top\mathbf{K}\cdot\begin{pmatrix}\mathbf{H}_{f',\mathbf{x}}\cdot\mathbf{u}\\ 1\end{pmatrix} \\
&\approx \left(\mathbf{s}_0'^\top(\mathbf{A}_{f'}\mathbf{u}\otimes\mathbf{I}_m) + \mathbf{s}_1'^\top\mathbf{B}_0\right)\cdot\mathbf{r} - \mathbf{s}_0'^\top(\mathbf{A}_{f'}\mathbf{u}\otimes\mathbf{I}_m)\cdot\mathbf{r} - \mathbf{s}_1'^\top\mathbf{B}_0\mathbf{r} \\
&= \mu\cdot\mathbf{g}^\top\mathbf{r}
\end{aligned}
$$

The decryption noise is as follows:

$$
e_{\mathsf{dec}} = \mathbf{e}'^\top\mathbf{r} + \mathbf{e}_0'^\top\mathbf{K}\cdot\begin{pmatrix}\mathbf{H}_{f',\mathbf{x}}\cdot\mathbf{u}\\ 1\end{pmatrix}.
$$

The bound on the noise term in the updated ciphertexts increases with each update. To represent the number of updates, we use superscripts (i). Note that (0) denotes fresh ciphertexts.

- $\mathbf{e}_0^{(0)} \sim \mathcal{D}_{\mathbb{Z}^{m'},\sigma}$, $\mathbf{e}_0^{(i)} \sim \mathcal{D}_{\mathbb{Z}^{m'},\sigma\sqrt{m''/2}}$ for $i > 0$
- $\mathbf{e}^{(0)} \sim \mathcal{D}_{\mathbb{Z}^m,\sigma}$
  - $\mathbf{e}^{(1)} = \mathbf{e}^{(0)} + \mathbf{E}\mathbf{G}^{-1}(\beta_0^{(0)}) - \mathbf{R}^\top\mathbf{e}_0^{(0)}$
  - $\mathbf{e}^{(i)} = \mathbf{e}^{(i-1)} + \underbrace{\mathbf{E}\mathbf{G}^{-1}(\beta_0^{(i-1)}) - \mathbf{R}^\top\mathbf{e}_0^{(i-1)}}_{\mathbf{e}_{\mathsf{inc}}}$

The error $\mathbf{e}^{(Q)}$ in uct is bounded by

$$
\begin{aligned}
&|\mathbf{e}^{(1)}| + (Q-1)|\mathbf{e}_{\mathsf{inc}}| \\
&\le \sigma\sqrt{m}\cdot\lambda^{\omega(1)} + \sqrt{m}\cdot\sigma\cdot\sqrt{\frac{m''}{2}}\cdot\lambda^{\omega(1)} + (\sqrt{m'}+\sqrt{m})\sigma_{-1}\cdot\sqrt{m'}\cdot\sigma\cdot\lambda^{\omega(1)} \\
&\quad + Q\left(\sqrt{m}\sigma\sqrt{\frac{m''}{2}}\cdot\lambda^{\omega(1)} + (\sqrt{m'}+\sqrt{m})\sigma_{-1}\cdot\sqrt{m'}\cdot\sigma\cdot\sqrt{\frac{m''}{2}}\cdot\lambda^{\omega(1)}\right) \\
&\le Q\cdot O\left(\sigma\sqrt{\frac{m''}{2}}(\sqrt{m} + (m+\sqrt{m\cdot m'})\cdot\sigma_{-1})\right)\lambda^{\omega(1)},
\end{aligned}
$$

where Q is the maximum number of updates.

When $|e^{(Q)}| < B$, the total decryption error is bounded by

$$\begin{aligned}
&\left\|\mathbf{e}^{(Q)}\right\| \|\mathbf{r}\| + \left\|\mathbf{e}_0^{(Q)}\right\| (\|\mathbf{K}_0\| \cdot \|\mathbf{H}_{f',\mathbf{x}}\| \cdot \|\mathbf{u}\| - \|\mathbf{k}_1\|) \\
&\leq B\sigma'\sqrt{m}\lambda^{\omega(1)} + \sigma\sqrt{\frac{m''}{2}}\sqrt{m'}((\sqrt{m'} + \sqrt{\ell m})\sigma_{-1}m^{O(d)}\sqrt{m} \cdot \sigma' + \sqrt{m'}\sigma_{-1})\lambda^{\omega(1)} \\
&\leq B\sigma'\sqrt{m}\lambda^{\omega(1)} + \sigma\sqrt{\frac{m''}{2}} \cdot O(\sqrt{m'} \cdot \sigma_{-1} \cdot \sigma') \cdot m^{O(d)}\lambda^{\omega(1)} \leq \gamma,
\end{aligned}$$

where $\mathbf{K}_0$ denote all but the last columns of $\mathbf{K}$ and $\mathbf{k}_1$ is the last column of $\mathbf{K}$. Therefore, correctness holds with our parameters setting. □

Security.

Theorem 5. *Assuming the hardness of public-coin evasive LWE (Definition 3), tensor LWE (Definition 4) and LWE (Definition 1), the updatable CP-ABE scheme in Construction 4 is very selectively secure.*

The high-level proof strategy follows a similar structure to that of Theorem 4. First, leveraging the public-coin evasive LWE and tensor LWE assumptions, we transform the challenge ciphertext and all update tokens whose after-update attributes appear in the update chain involving the challenge policy f^* into uniformly random. Subsequently, all honestly generated updated ciphertexts derived from the challenge ciphertext ct^* are similarly replaced with uniformly random.

Proof. Suppose that the adversary $\mathcal{A}$ with randomness $\mathsf{coin}_{\mathcal{A}}$ submits the challenge policy f^*, secret key queries $\mathcal{K} = \{\mathbf{x}_k\}_{k\in[Q_{\mathsf{sk}}]}$, update token queries $\mathcal{CT} = \{f_{i,\text{before}} \to f_{i,\text{after}}\}_{i\in[Q_{\mathsf{ct}}]}$ and updating ciphertext queries $\mathcal{D} = \{f_{j,\text{before}} \to f_{j,\text{after}}\}_{j\in[Q_{\mathsf{uct}}]}$ at the very beginning.

From the update token queries, we define a subset $\mathcal{CT}^* = \{f^*_{i,\text{before}} \to f^*_{i,\text{after}}\}_{i\in[Q^*_{\mathsf{ct}}]}$ containing all policies that appear in the update chain starting from the challenge policy f^*. Similarly, from the updating ciphertext queries, we derive a subset $\mathcal{D}^* = \{f^* \to f^*_j\}_{j\in[Q^*_{\mathsf{uct}}]}$, which consists of all updates where the before-update ciphertext corresponds to the challenge ciphertext ct^*. Define a set $\mathcal{UC} := f^* \cup \mathcal{D}^* \cup \mathcal{CT}^*$ and denote $|\mathcal{UC}|$ by Q_{uc}.

In order to rely on $\mathsf{PublicEvLWE}$ assumption, we consider the pre- and post-experiment $\mathsf{pre}^b_{\mathcal{B}}(\lambda)$ and $\mathsf{post}^b_{\mathcal{A}}(\lambda)$, as showed in Figure 8 and Figure 9. In both experiments, the grayboxed components are chosen from corresponding uniformly random distribution. The security of CU-CP-ABE scheme explicitly follows from the indistinguishability of $\mathsf{post}^b_{\mathcal{A}}(\lambda)$, which implies that the challenge ciphertext, update tokens in $\mathcal{CT}^*$ and updated ciphertexts in $\mathcal{D}^*$ are jointly distributed close to uniformly random. By relying on the public-coin evasive LWE assumption, it suffices to prove the pseudorandomness of the output of $\mathsf{pre}^b_{\mathcal{B}}(\lambda)$.

The proof proceeds via a sequence of hybrids $\mathsf{G}^b_0, \mathsf{G}^b_1, \mathsf{G}^b_2, \mathsf{G}^b_3$ for $b \in \{0,1\}$.

G^b_0 **for** $b \in \{0,1\}$**:** This is the original experiment for challenge bit b.

G^b_1 **for** $b \in \{0,1\}$**:** The challenge ciphertext components are switched to random. To show that $\mathsf{G}^b_0 \approx_c \mathsf{G}^b_1$, we rewrite the view of the adversary $\mathcal{A}$ as

$$\begin{aligned}
&\mathsf{mpk} = (\mathbf{B}, \mathbf{B}_0, \mathbf{A}_{\mathsf{attr}}, \mathbf{u}), \\
&\beta^*_0 = \mathbf{B}^\top \begin{pmatrix} \mathbf{s}^*_0 \\ \mathbf{s}^*_1 \end{pmatrix} + \mathbf{e}^*_0,\ \{\beta'^*_{0,k} = ((\mathbf{A}_{\mathsf{attr}} + \mathbf{x}_k^\top \otimes \mathbf{G}) \otimes \mathbf{r}_k)^\top \mathbf{s}^*_0 + \mathbf{e}'_{0^*,k}\}_{k\in Q_{\mathsf{sk}}}, \\
&\{\beta'^*_{1,k} = (\mathbf{B}_0 \cdot \mathbf{r}_{\mathbf{k}})^\top \mathbf{s}^*_1 + e'_{1^*,k}\}_{k\in Q_{\mathsf{sk}}},\ \mathbf{c}'_{f^*} = (\mathbf{A}_{f^*}\mathbf{u} \otimes \mathbf{I}_m)^\top \mathbf{s}^*_0 + \mathbf{B}_0^\top \mathbf{s}^*_1 + \mathbf{e}^*.
\end{aligned}$$

Define the following intermediate hybrid games.

- $\mathsf{G}^b_{1,0}$: Same as G^b_0.
- $\mathsf{G}^b_{1,1}$: Same as $\mathsf{G}^b_{1,0}$, except we compute

$$\beta'^*_{1,k} = \mathbf{c}'^\top_{f^*} \cdot \mathbf{r}_k - (\beta'^*_{0,k})^\top \cdot \mathbf{H}_{f^*,\mathbf{x}_k} \cdot \mathbf{u} + \mathbf{s}^{*\top}_0 (\mathbf{G} \cdot \mathbf{u} \otimes \mathbf{r}_k) + e'_{1^*,k}.$$

$\mathsf{pre}^b_{\mathcal{B}}(\lambda)$

$\mathbf{B} \xleftarrow{\$} \mathbb{Z}_q^{nm \times m'}$

$(\{\mathbf{A}'_{\hat{i}}\}_{\hat{i} \in Q_{\mathsf{uc}}}, \mathbf{P}, \mathsf{aux}) \leftarrow \mathsf{Samp}(1^\lambda; \mathsf{rand})$

$\mathbf{H}_{f,\hat{i}} \leftarrow \mathsf{EvalF}(\mathbf{A}_{\mathsf{attr}}, f_{\hat{i}}), \mathbf{A}_{f,\hat{i}} \coloneqq \mathbf{A}_{\mathsf{attr}} \cdot \mathbf{H}_{f,\hat{i}}$ for all $f_{\hat{i}} \in \mathcal{UC}$ (denote f^* by $f_{\hat{i}^*}$)

$\mathbf{A}'_{\hat{i}} \coloneqq [(\mathbf{A}_{f_{\hat{i}}} \mathbf{u} \otimes \mathbf{I}_m)^\top | \mathbf{B}_0^\top]$

$\mathbf{P}_{0,k} = (\mathbf{A}_{\mathsf{attr}} - \mathbf{x}_k^\top \otimes \mathbf{G}) \otimes \mathbf{r}_k,\ \mathbf{p}_{1,k} = \mathbf{B}_0 \cdot \mathbf{r}_k$ for all $k \in [Q_{\mathsf{sk}}]$

$\mathbf{P} = \begin{pmatrix} \mathbf{P}_0 & \\ & \mathbf{P}_1 \end{pmatrix}, \mathbf{P}_0 = [\mathbf{P}_{0,1} | \dots | \mathbf{P}_{0,Q_{\mathsf{sk}}}], \mathbf{P}_1 = [\mathbf{p}_{1,1} | \dots | \mathbf{p}_{1,Q_{\mathsf{sk}}}]$

$\mathsf{aux} = (\mathbf{A}_{\mathsf{attr}}, \mathbf{B}_0, \mathbf{u}, \{\mathbf{r}_k\}_{k \in [Q_{\mathsf{sk}}]}, \mathsf{coin}_{\mathcal{A}}, f^*, \{f_j^*\}_{j \in [Q^*_{\mathsf{uct}}]}, \{f^*_{i,\mathrm{after}}\}_{i \in [Q^*_{\mathsf{ct}}]}, \{\mathbf{x}_k\}_{k \in [Q_{\mathsf{sk}}]})$

$\mathbf{s}_0^* \xleftarrow{\$} \mathbb{Z}_q^{nm}, \mathbf{s}_1^* \xleftarrow{\$} \mathbb{Z}_q^n, \mathbf{S}_{0,\hat{i}} \xleftarrow{\$} \mathbb{Z}_q^{nm \times m''}, \mathbf{S}_{1,\hat{i}} \xleftarrow{\$} \mathbb{Z}_q^{n \times m''}$

if $b = 0$ **then**

$\mathbf{e}_0^* \leftarrow \mathcal{D}_{\mathbb{Z}^{m'},\sigma}, \mathbf{e} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma}, \mathbf{e}_{0^*,k} \leftarrow \mathcal{D}_{\mathbb{Z}^{m'},\sigma''}, e_{1^*,k} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma},$

$\beta_0^* \coloneqq \mathbf{B}^\top \begin{pmatrix} \mathbf{s}_0^* \\ \mathbf{s}_1^* \end{pmatrix} + \mathbf{e}_0^*,\ \mathbf{c}'_{f^*} \coloneqq (\mathbf{A}_{f^*} \mathbf{u} \otimes \mathbf{I}_m)^\top \mathbf{s}_0^* + \mathbf{B}_0^\top \mathbf{s}_1^* + \mathbf{e}^*,$

$\left\{\beta'^*_{0,k} \coloneqq \mathbf{P}_{0,k}^\top \mathbf{s}_0^* + \mathbf{e}_{0^*,k},\ \beta'^*_{1,k} \coloneqq \mathbf{p}_{1,k}^\top \mathbf{s}_1^* + e_{1^*,k}\right\}_{k \in Q_{\mathsf{sk}}}$

$\mathbf{E}_{0,\hat{i}} \leftarrow \mathcal{D}_{\mathbb{Z}^{m' \times m''},\sigma}, \mathbf{E}_{\hat{i}} \leftarrow \mathcal{D}_{\mathbb{Z}^{m \times m''},\sigma}, \mathbf{E}'_{0,\hat{i},k} \leftarrow \mathcal{D}_{\mathbb{Z}^{m' \times m''},\sigma''}, \mathbf{e}'_{1,\hat{i},k} \leftarrow \mathcal{D}_{\mathbb{Z}^{m \times m''},\sigma},$

$\left\{\mathbf{D}_{0,\hat{i}} \coloneqq \mathbf{B}^\top \begin{pmatrix} \mathbf{S}_{0,\hat{i}} \\ \mathbf{S}_{1,\hat{i}} \end{pmatrix} + \mathbf{E}_{0,\hat{i}}, \mathbf{D}'_{f,\hat{i}} \coloneqq (\mathbf{A}_{f_{\hat{i}}} \mathbf{u} \otimes \mathbf{I}_m)^\top \mathbf{S}_{0,\hat{i}} + \mathbf{B}_0^\top \mathbf{S}_{1,\hat{i}} + \mathbf{E}_{\hat{i}}, \right\}_{\hat{i} \in [Q_{\mathsf{uc}}] \setminus \hat{i}^*}$

$\left\{\mathbf{D}'_{0,\hat{i},k} \coloneqq \mathbf{P}_{0,k}^\top \mathbf{S}_{0,\hat{i}} + \mathbf{E}'_{0,\hat{i},k}, \mathbf{D}'_{1,\hat{i},k} \coloneqq \mathbf{p}_{1,k}^\top \mathbf{S}_{1,\hat{i}} + \mathbf{e}'_{1,\hat{i},k}\right\}_{\hat{i} \in [Q_{\mathsf{uc}}] \setminus \hat{i}^*, k \in [Q_{\mathsf{sk}}]}$

if $b = 1$ **then**

$\beta_0^*, \{\beta'^*_{0,k}, \beta'^*_{1,k}\}_{k \in Q_{\mathsf{sk}}}, \mathbf{c}'_{f^*}, \left\{\mathbf{D}_{0,\hat{i}}, \mathbf{D}'_{0,\hat{i},k}, \mathbf{D}'_{1,\hat{i},k}, \mathbf{D}'_{f,\hat{i}}\right\}_{\hat{i} \in [Q_{\mathsf{uc}}] \setminus \hat{i}^*, k \in [Q_{\mathsf{sk}}]}$

return $\begin{pmatrix} \mathbf{B}, & \{\mathbf{A}'_{\hat{i}}\}_{\hat{i} \in Q_{\mathsf{uc}}}, \\ \beta_0'^*, \{\mathbf{D}_{0,\hat{i}}\}_{\hat{i} \in [Q_{\mathsf{uc}}] \setminus \hat{i}^*}, & \mathbf{c}'_{f^*}, \{\mathbf{D}'_{f,\hat{i}}\}_{\hat{i} \in [Q_{\mathsf{uc}}] \setminus \hat{i}^*}, \\ & \mathbf{P}, \\ \{\beta'^*_{0,k}, \beta'^*_{1,k}, \mathbf{D}'_{0,\hat{i},k}, \mathbf{D}'_{1,\hat{i},k}\}_{\hat{i} \in [Q_{\mathsf{uc}}] \setminus \hat{i}^*, k \in [Q_{\mathsf{sk}}]}, & \mathsf{aux} \end{pmatrix}$

(In the above experiment, $\mathbf{A}_{\mathsf{attr}}, \mathbf{B}_0, \mathbf{u}, \{\mathbf{r}_k\}_{k \in [Q_{\mathsf{sk}}]}$ are sampled as in normal scheme.)

Fig. 8. Experiment $\mathsf{pre}^b_{\mathcal{B}}(\lambda)$ for PublicEvLWE.

$\mathsf{post}^b_{\mathcal{A}}(\lambda)$

$1: \mathbf{B} \xleftarrow{\$} \mathbb{Z}_q^{nm \times m'}$

$2: (\{\mathbf{A}'_{\hat{i}}\}_{\hat{i} \in Q_{\mathsf{uc}}}, \mathbf{P}, \mathsf{aux}) \leftarrow \mathsf{Samp}(1^\lambda; \mathsf{rand})$ as in $\mathsf{pre}^b_{\mathcal{B}}(\lambda)$

$3: \mathbf{s}_0^* \xleftarrow{\$} \mathbb{Z}_q^{nm}, \mathbf{s}_1^* \xleftarrow{\$} \mathbb{Z}_q^{n}, \mathbf{S}_{0,\hat{i}} \xleftarrow{\$} \mathbb{Z}_q^{nm \times m''}, \mathbf{S}_{1,\hat{i}} \xleftarrow{\$} \mathbb{Z}_q^{n \times m''}$

$4:$ **if** $b = 0$ **then**

$5:$ Sample $\mathbf{e}_0^*, \mathbf{e}$ as in $\mathsf{pre}^b_{\mathcal{B}}(\lambda)$

$$\beta_0^* \coloneqq \mathbf{B}^\top \begin{pmatrix} \mathbf{s}_0^* \\ \mathbf{s}_1^* \end{pmatrix} + \mathbf{e}_0^*, \ \mathbf{c}'_{f^*} \coloneqq (\mathbf{A}_{f^*}\mathbf{u} \otimes \mathbf{I}_m)^\top \mathbf{s}_0^* + \mathbf{B}_0^\top \mathbf{s}_1^* + \mathbf{e}^*,$$

$6:$ Sample $\mathbf{E}_{0,\hat{i}}, \mathbf{E}_{\hat{i}}$ as in $\mathsf{pre}^b_{\mathcal{B}}(\lambda)$

$$\left\{ \mathbf{D}_{0,\hat{i}} \coloneqq \mathbf{B}^\top \begin{pmatrix} \mathbf{S}_{0,\hat{i}} \\ \mathbf{S}_{1,\hat{i}} \end{pmatrix} + \mathbf{E}_{0,\hat{i}}, \mathbf{D}'_{f,\hat{i}} \coloneqq (\mathbf{A}_{f_{\hat{i}}}\mathbf{u} \otimes \mathbf{I}_m)^\top \mathbf{S}_{0,\hat{i}} + \mathbf{B}_0^\top \mathbf{S}_{1,\hat{i}} + \mathbf{E}_{\hat{i}}, \right\}_{\hat{i} \in [Q_{\mathsf{uc}}] \setminus \hat{i}^*}$$

$7:$ $$\left\{ \mathbf{K}_k \coloneqq \mathbf{B}^{-1} \begin{pmatrix} (\mathbf{A}_{\mathsf{attr}} + \mathbf{x}_k^\top \otimes \mathbf{G}) \otimes \mathbf{r}_k \\ \mathbf{B}_0 \mathbf{r}_k \end{pmatrix} \right\}_{k \in [Q_{\mathsf{sk}}]}$$

$8:$ **if** $b = 1$ **then**

$9:$ $$\boxed{\beta_0^*}, \boxed{\mathbf{c}'_{f^*}}, \left\{ \boxed{\mathbf{D}_{0,\hat{i}}}, \boxed{\mathbf{D}'_{f,\hat{i}}} \right\}_{\hat{i} \in [Q_{\mathsf{uc}}] \setminus \hat{i}^*},$$

$10:$ $$\left\{ \mathbf{K}_k \coloneqq \mathbf{B}^{-1} \begin{pmatrix} (\mathbf{A}_{\mathsf{attr}} + \mathbf{x}_k^\top \otimes \mathbf{G}) \otimes \mathbf{r}_k \\ \mathbf{B}_0 \mathbf{r}_k \end{pmatrix} \right\}_{k \in [Q_{\mathsf{sk}}]}$$

$11:$ **return** $$\begin{pmatrix} \mathbf{B}, & \{\mathbf{A}'_{\hat{i}}\}_{\hat{i} \in Q_{\mathsf{uc}}}, \\ \beta_0'^*, \{\mathbf{D}_{0,\hat{i}}\}_{\hat{i} \in [Q_{\mathsf{uc}}] \setminus \hat{i}^*}, & \mathbf{c}'_{f^*}, \{\mathbf{D}'_{f,\hat{i}}\}_{\hat{i} \in [Q_{\mathsf{uc}}] \setminus \hat{i}^*}, \\ \mathbf{P}, & \\ \{\mathbf{K}_k\}_{k \in [Q_{\mathsf{sk}}]}, & \mathsf{aux} \end{pmatrix}$$

(In the above experiment, $\mathbf{A}_{\mathsf{attr}}, \mathbf{B}_0, \mathbf{u}, \{\mathbf{r}_k\}_{k \in [Q_{\mathsf{sk}}]}$ are sampled as in normal scheme.)

Fig. 9. Experiment $\mathsf{post}^b_{\mathcal{A}}(\lambda)$ for PublicEvLWE.

We observe that

$$\begin{aligned} & \mathbf{s}_1^{*\top}(\mathbf{B}_0 \cdot \mathbf{r}_k) \\ = \ & \underbrace{(\mathbf{s}_0^{*\top}(\mathbf{A}_{f^*}\mathbf{u} \otimes \mathbf{I}_m) + \mathbf{s}_1^{*\top}\mathbf{B}_0)}_{\approx \mathbf{c}'^\top_{f^*}} \cdot \mathbf{r}_k - \mathbf{s}_0^{*\top}(\mathbf{A}_{f^*}\mathbf{u} \otimes \mathbf{r}_k) \\ = \ & \mathbf{c}'^\top_{f^*} \cdot \mathbf{r}_k - \underbrace{\mathbf{s}_0^{*\top}((\mathbf{A}_{\mathsf{attr}} + \mathbf{x}_k^\top \otimes \mathbf{G}) \otimes \mathbf{r}_k)}_{\approx (\beta'^*_{0,k})^\top} \cdot \mathbf{H}_{f^*,\mathbf{x}_k} \cdot \mathbf{u} + \mathbf{s}_0^{*\top}(\mathbf{G} \cdot \mathbf{u} \otimes \mathbf{r}_k). \end{aligned}$$

Using $e'_{1^*,k}$ for noise flooding, we have

$$e'_{1^*,k} \approx_s e'_{1^*,k} + \mathbf{e}^{*\top} \cdot \mathbf{r}_k + e^\top_{0^*,k} \cdot \mathbf{H}_{f^*,\mathbf{x}_k} \cdot \mathbf{u},$$

where $e_{1^*,k} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}$ and $\sigma > m^{O(d)} \cdot \lambda^{\omega(1)}$. Hence, $\mathsf{G}^b_{1,1} \approx_s \mathsf{G}^b_{1,0}$.

– $\mathsf{G}^b_{1,2}$: Same as $\mathsf{G}^b_{1,1}$, except we sample $\beta_0^*, \mathbf{c}'_{f^*}$ from corresponding uniformly random distributions. We have

$$(\mathbf{B}_{\mathsf{bot}}, \mathbf{B}_0, \mathbf{B}_{\mathsf{bot}}^\top \mathbf{s}_1^* + \mathbf{e}_0^*, \mathbf{B}_0^\top \mathbf{s}_1^* + \mathbf{e}^*) \approx_c (\mathbf{B}_{\mathsf{bot}}, \mathbf{B}_0, \boxed{\beta_0^*}, \boxed{\mathbf{c}''_{f^*}}),$$

where $\mathbf{B}_{\mathsf{bot}}$ denotes the last n rows of $\mathbf{B}$ and the components in gray box are chosen from uniformly random. Therefore, $\mathsf{G}^b_{1,2} \approx_c \mathsf{G}^b_{1,1}$ follows from the LWE hardness assumption.

– $\mathsf{G}^b_{1,3}(\equiv \mathsf{G}^b_1)$: Same as $\mathsf{G}^b_{1,2}$, except we sample $\beta'^*_{0,k}, \beta'^*_{1,k}$ from corresponding uniformly random distributions. We switch these components to random step by step as follows:

Using the mixed-product property, we can rewrite $\beta'^*_{0,k}$ (apart from its noise) as follows:

$$\begin{aligned}&\mathbf{s}_0^{*\top}((\mathbf{A}_{\mathsf{attr}} + \mathbf{x}_k^\top \otimes \mathbf{G}) \otimes \mathbf{r}_k)\\ =\ &\mathbf{s}_0^{*\top}(\mathbf{I}_n \cdot (\mathbf{A}_{\mathsf{attr}} + \mathbf{x}_k^\top \otimes \mathbf{G}) \otimes \mathbf{r}_k \cdot 1)\\ =\ &\mathbf{s}_0^{*\top}(\mathbf{I}_n \otimes \mathbf{r}_k)(\mathbf{A}_{\mathsf{attr}} + \mathbf{x}_k^\top \otimes \mathbf{G}).\end{aligned}$$

Similarly, we have $\mathbf{s}_0^{*\top}(\mathbf{G} \cdot \mathbf{u} \otimes \mathbf{r}_k) = \mathbf{s}_0^{*\top}(\mathbf{I}_n \otimes \mathbf{r}_k) \cdot \mathbf{G}\mathbf{u}$. Relying on the tensor LWE assumption, we directly have

$$\begin{aligned}&\Big\{\mathbf{A}_{\mathsf{attr}}, \Big\{\mathbf{r}_k, \begin{pmatrix}\mathbf{s}_0^{*\top}(\mathbf{I}_n \otimes \mathbf{r}_k)(\mathbf{A}_{\mathsf{attr}} + \mathbf{x}_k \otimes \mathbf{G}) + \mathbf{e}^\top_{0^*,k},\\ \mathbf{s}_0^{*\top}(\mathbf{I}_n \otimes \mathbf{r}_k) \cdot \mathbf{G}\mathbf{u} + e_{1^*,k}\end{pmatrix}\Big\}_{k\in[Q_{\mathsf{sk}}]}\Big\}\\ \approx_c\ &\Big\{\mathbf{A}_{\mathsf{attr}}, \{\mathbf{r}_k, \beta'^*_{0,k}, \beta''^*_{1,k}\}_{k\in[Q_{\mathsf{sk}}]}\Big\}\ ,\end{aligned}$$

where $\beta'^*_{0,k}$ and $\beta''^*_{1,k}$ are sampled uniform randomly from $\mathbb{Z}_q^{m'}$ and $\mathbb{Z}_q$, respectively. Therefore, $\beta'^*_{1,k}$ is statistically close to uniformly random as $\mathbf{s}_0^{*\top}(\mathbf{I}_n \otimes \mathbf{r}_k) \cdot \mathbf{G}\mathbf{u} + e_{1^*,k}$ has been switched to random.

G^b_2 **for** $b \in \{0,1\}$**:** The components of updated tokens for queries with index $i \in Q^*_{\mathsf{uct}} \cup Q^*_{\mathsf{ct}}$ are switched to random.

Similar to the proof of Theorem 4, the gap of G^b_2 and G^b_1 can be separated into a series of intermediate hybrids where each pair of adjacent hybrids are statistically close follows from the fact that $\mathsf{G}^b_0 \approx_c \mathsf{G}^b_1$.

G^b_3 **for** $b \in \{0,1\}$**:** The components of updated ciphertexts that starting from the challenge ciphertext are switched to random. At the end of G^b_2, since all the inputs required for computing update ciphertexts $\{\mathsf{uct}_{\hat{i}}\}_{\hat{i}\in Q^*_{\mathsf{uct}}}$ have been switched to random, the resulting $\mathsf{uct}_{\hat{i}}$ are distributed as random, as in G^b_3.

□

Acknowledgements

The first author's work was supported by EPSRC grant EP/Y02432X/1. The third author's work was supported by the Austrian Science Fund (FWF) Project J4879-N, by the Bakar Funds and Peder Sather Funds. The fourth author was supported in part by the National Key R&D Program of China under Grant No. 2023YFB4403500, in part by the National Natural Science Foundation of China (Grants No. 62272362 and No. 12441104), the National Cryptologic Science Fund of China (Grant No. 2025NCSFO1002), and the Youth Innovation Team of Shaanxi Universities.

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, Berlin, Heidelberg, May / June 2010.

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, Cham, December 2020.

ACJ17. Prabhanjan Ananth, Aloni Cohen, and Abhishek Jain. Cryptography with updates. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 445–472. Springer, Cham, April / May 2017.

Agr19. Shweta Agrawal. Indistinguishability obfuscation without multilinear maps: New methods for bootstrapping and instantiation. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 191–225. Springer, Cham, May 2019.

AKY24. Shweta Agrawal, Simran Kumari, and Shota Yamada. Compact pseudorandom functional encryption from evasive LWE. Cryptology ePrint Archive, Report 2024/1719, 2024.

ALMT20. Shweta Agrawal, Benoît Libert, Monosij Maitra, and Radu Titiu. Adaptive simulation security for inner product functional encryption. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part I*, volume 12110 of *LNCS*, pages 34–64. Springer, Cham, May 2020.

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, Berlin, Heidelberg, August 2016.

AMYY25. Shweta Agrawal, Anuja Modi, Anshu Yadav, and Shota Yamada. Zeroizing attacks against evasive and circular evasive LWE. In Benny Applebaum and Huijia (Rachel) Lin, editors, *TCC 2025, Part II*, volume 16269 of *LNCS*, pages 259–290. Springer, Cham, December 2025.

AP20. Shweta Agrawal and Alice Pellet-Mary. Indistinguishability obfuscation without maps: Attacks and fixes for noisy linear FE. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 110–140. Springer, Cham, May 2020.

AV19. Prabhanjan Ananth and Vinod Vaikuntanathan. Optimal bounded-collusion secure functional encryption. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part I*, volume 11891 of *LNCS*, pages 174–198. Springer, Cham, December 2019.

AYY22. Shweta Agrawal, Anshu Yadav, and Shota Yamada. Multi-input attribute based encryption and predicate encryption. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 590–621. Springer, Cham, August 2022.

BGG+14. Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 533–556. Springer, Berlin, Heidelberg, May 2014.

BLMR13. Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic PRFs and their applications. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 410–428. Springer, Berlin, Heidelberg, August 2013.

BP14. Abhishek Banerjee and Chris Peikert. New and improved key-homomorphic pseudorandom functions. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 353–370. Springer, Berlin, Heidelberg, August 2014.

BPR12. Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 719–737. Springer, Berlin, Heidelberg, April 2012.

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

BÜW24. Chris Brzuska, Akin Ünal, and Ivy K. Y. Woo. Evasive LWE assumptions: Definitions, classes, and counterexamples. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part IV*, volume 15487 of *LNCS*, pages 418–449. Springer, Singapore, December 2024.

BV15. Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In Venkatesan Guruswami, editor, *56th FOCS*, pages 171–190. IEEE Computer Society Press, October 2015.

CRS+23. Valerio Cini, Sebastian Ramacher, Daniel Slamanig, Christoph Striecks, and Erkan Tairi. (inner-product) functional encryption with updatable ciphertexts. *Journal of Cryptology*, 37(1):8, Dec 2023.

DGP22. Cécile Delerablée, Lénaïck Gouriou, and David Pointcheval. Key-policy ABE with switchable attributes. In Clemente Galdi and Stanislaw Jarecki, editors, *SCN 22*, volume 13409 of *LNCS*, pages 147–171. Springer, Cham, September 2022.

DHM+24. Fangqi Dong, Zihan Hao, Ethan Mook, Hoeteck Wee, and Daniel Wichs. Laconic function evaluation and ABE for RAMs from (ring-)LWE. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part III*, volume 14922 of *LNCS*, pages 107–142. Springer, Cham, August 2024.

DJM+25. Nico Döttling, Abhishek Jain, Giulio Malavolta, Surya Mathialagan, and Vinod Vaikuntanathan. Simple and general counterexamples for private-coin evasive LWE. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part VII*, volume 16006 of *LNCS*, pages 73–92. Springer, Cham, August 2025.

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, Berlin, Heidelberg, August 2014.

Fug21. Somchart Fugkeaw. A lightweight policy update scheme for outsourced personal health records sharing. *IEEE Access*, 9:54862–54871, 2021.

GKPV10. Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Robustness of the learning with errors assumption. In Andrew Chi-Chih Yao, editor, *ICS 2010*, pages 230–240. Tsinghua University Press, January 2010.

GKVW20. Rishab Goyal, Venkata Koppula, Satyanarayana Vusirikala, and Brent Waters. On perfect correctness in (lockable) obfuscation. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 229–259. Springer, Cham, November 2020.

GKW16. Rishab Goyal, Venkata Koppula, and Brent Waters. Semi-adaptive security and bundling functionalities made generic and easy. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 361–388. Springer, Berlin, Heidelberg, October / November 2016.

GKW17. Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In Chris Umans, editor, *58th FOCS*, pages 612–621. IEEE Computer Society Press, October 2017.

GPSW06. Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 89–98. ACM Press, October / November 2006. Available as Cryptology ePrint Archive Report 2006/309.

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.

GWMM23. Lifeng Guo, Lingxia Wang, Xueke Ma, and Qianli Ma. A new revocable attribute based encryption on lattice. In Mingwu Zhang, Man Ho Au, and Yudi Zhang, editors, *Provable and Practical Security*, pages 309–326, Cham, 2023. Springer Nature Switzerland.

HJL25. Yao-Ching Hsieh, Aayush Jain, and Huijia Lin. Lattice-based post-quantum iO from circular security with random opening assumption. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part VII*, volume 16006 of *LNCS*, pages 3–38. Springer, Cham, August 2025.

HLL23a. Yao-Ching Hsieh, Huijia Lin, and Ji Luo. Attribute-based encryption for circuits of unbounded depth from lattices. In *64th FOCS*, pages 415–434. IEEE Computer Society Press, November 2023.

HLL23b. Yao-Ching Hsieh, Huijia Lin, and Ji Luo. Attribute-based encryption for circuits of unbounded depth from lattices. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 415–434. IEEE, 2023.

Jia20. Yao Jiang. The direction of updatable encryption does not matter much. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 529–558. Springer, Cham, December 2020.

JSMG18. Yinhao Jiang, Willy Susilo, Yi Mu, and Fuchun Guo. Ciphertext-policy attribute-based encryption supporting access policy update and its extension with preserved attributes. *International Journal of Information Security*, 17:533–548, 2018.

KNYY20. Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Adaptively secure inner product encryption from LWE. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 375–404. Springer, Cham, December 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, Berlin, Heidelberg, December 2016.

LAKW+23. Fucai Luo, Saif Al-Kuwari, Haiyan Wang, Fuqun Wang, and Kefei Chen. Revocable attribute-based encryption from standard lattices. *Computer Standards & Interfaces*, 84:103698, 2023.

LAKWC21. Fucai Luo, Saif Al-Kuwari, Fuqun Wang, and Kefei Chen. Attribute-based proxy re-encryption from standard lattices. *Theoretical Computer Science*, 865:52–62, 2021.

LCLS09. Xiaohui Liang, Zhenfu Cao, Huang Lin, and Jun Shao. Attribute based proxy re-encryption with delegating capabilities. In *Proceedings of the 4th international symposium on information, computer, and communications security*, pages 276–286, 2009.

LFSW13. Kaitai Liang, Liming Fang, Willy Susilo, and Duncan S. Wong. A ciphertext-policy attribute-based proxy re-encryption with chosen-ciphertext security. In *2013 5th International Conference on Intelligent Networking and Collaborative Systems*, pages 552–559, 2013.

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, Cham, October 2021.

LWYW24. Fucai Luo, Haiyan Wang, Xingfu Yan, and Jiahui Wu. Key-policy attribute-based encryption with switchable attributes for fine-grained access control of encrypted data. *IEEE Transactions on Information Forensics and Security*, 2024.

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, Berlin, 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.

Nis22. Ryo Nishimaki. The direction of updatable encryption does matter. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022, Part II*, volume 13178 of *LNCS*, pages 194–224. Springer, Cham, March 2022.

PO22. Harsha Sandaruwan Gardiyawasam Pussewalage and Vladimir Oleshchuk. A delegatable attribute based encryption scheme for a collaborative e-health cloud. *IEEE Transactions on Services Computing*, 16(2):787–801, 2022.

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.

SDDR21. Willy Susilo, Priyanka Dutta, Dung Hoang Duong, and Partha Sarathi Roy. Lattice-based hra-secure attribute-based proxy re-encryption in standard model. In Elisa Bertino, Haya Shulman, and Michael Waidner, editors, *Computer Security – ESORICS 2021*, pages 169–191, Cham, 2021. Springer International Publishing.

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, Berlin, Heidelberg, May 2005.

TKN20. Junichi Tomida, Yuto Kawahara, and Ryo Nishimaki. Fast, compact, and expressive attribute-based encryption. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part I*, volume 12110 of *LNCS*, pages 3–33. Springer, Cham, May 2020.

Tsa19. Rotem Tsabary. Fully secure attribute-based encryption for t-CNF from LWE. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 62–85. Springer, Cham, August 2019.

Tsa22. Rotem Tsabary. Candidate witness encryption from lattice techniques. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 535–559. Springer, Cham, August 2022.

Wee22. Hoeteck Wee. Optimal broadcast encryption and CP-ABE from evasive lattice assumptions. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 217–241. Springer, Cham, May / June 2022.

WFL19. Zhedong Wang, Xiong Fan, and Feng-Hao Liu. FE for inner products and its application to decentralized ABE. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part II*, volume 11443 of *LNCS*, pages 97–127. Springer, Cham, April 2019.

WZ17. Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under LWE. In Chris Umans, editor, *58th FOCS*, pages 600–611. IEEE Computer Society Press, October 2017.

YCJ23. Feng Yang, Hui Cui, and Jiwu Jing. Generic constructions of server-aided revocable abe with verifiable transformation. In *Applied Cryptography and Network Security Workshops*, pages 459–479, Cham, 2023. Springer Nature Switzerland.

YJR+14. Kan Yang, Xiaohua Jia, Kui Ren, Ruitao Xie, and Liusheng Huang. Enabling efficient access control with dynamic policy updating for big data in the cloud. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pages 2013–2021. IEEE, 2014.

A IPFE Schemes with Simulation-Based Security

In this section, we propose an IB-IPFE scheme with simulation-based security and an IPFE scheme with simulation-based security allowing a single key query and a bounded number of encryption queries. The first construction is needed for the generic construction in Section 4.1, the latter for the CU-PE scheme in Section 4.2.

A.1 Identity-Based Inner-Product Functional Encryption Scheme

In this section, we present an IB-IPFE scheme with simulation-based security, achieved by carefully combining techniques from IND-secure IB-IPFE schemes [ACGU20] and the SIM-secure IPFE construction [ALMT20]. Since the proposed IB-IPFE will be employed for generating at most Q update tokens in Construction 1, we focus on demonstrating the (Q, poly)-IB-IPFE variant here.

Construction 5 ((Q, poly)-IB-IPFE with SIM-based security).

$\mathsf{Setup}(1^\lambda, \mathcal{ID}, \mathcal{IP}, Q)$ *takes as input the security parameter* 1^λ*, the identity space* $\mathcal{ID}$ *and the class* $\mathcal{IP}$ *of inner product function indexed by vectors in* $\mathbb{Z}_p^\ell$*,*

1. *Sample* $(\mathbf{A}, \mathbf{T_A}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$.
2. *Choose random matrices* $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$ *and* $\mathbf{P} \xleftarrow{\$} \mathbb{Z}_q^{n\times(\ell+Q+1)}$.
3. *Output the public and master secret keys*

$$\mathsf{mpk} := (\mathbf{A}, \mathbf{B}, \mathbf{P}), \mathsf{msk} := \mathbf{T_A}.$$

$\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}, \mathbf{v})$ *takes as input* msk*, the identity* id *and key vector* $\mathbf{v}$*,*

1. *Compute* $\mathbf{H}_{\mathsf{id}} := \mathbf{B} + \mathsf{H}(\mathsf{id})$.
2. *Sample* $\mathbf{K}_{\mathsf{id}} \leftarrow \mathsf{SampleLeft}(\mathbf{A}, \mathbf{H}_{\mathsf{id}}, \mathbf{T_A}, \mathbf{P}, s)$*, s.t.* $[\mathbf{A}|\mathbf{H}_{\mathsf{id}}] \cdot \mathbf{K}_{\mathsf{id}} = \mathbf{P} \mod q$.
3. *Sample a random index* $t \xleftarrow{\$} [Q]$*, set* $\boldsymbol{\epsilon} \in \{0,1\}^Q$ *such that* $\boldsymbol{\epsilon}[t] = 1$ *and* $\boldsymbol{\epsilon}[i] = 0$ *for* $i \neq t$.
4. *Set* $\mathbf{v}' = (\mathbf{v}, \boldsymbol{\epsilon}, r) \in \mathbb{Z}_p^{\ell+Q+1}$ *for* $r \xleftarrow{\$} \mathbb{Z}_p$.
5. *Output* $\mathsf{sk} := (\mathbf{v}', \mathbf{K}_{\mathsf{id}} \cdot \mathbf{v}')$.

$\mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, \mathbf{u})$ *takes as input* mpk*, an identity* id *and a message* $\mathbf{u}$*,*

1. *Sample* $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\mathbf{e}_0 \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma}$, $\mathbf{e}_1 \leftarrow \mathcal{D}_{\mathbb{Z}^{\ell+Q+1},\sigma}$ *and* $\mathbf{e}_2 \leftarrow \mathcal{D}_{\mathbb{Z}^{\ell+Q+1},\sigma'}$.
2. *Compute* $\mathbf{H}_{\mathsf{id}} := \mathbf{B} + \mathsf{H}(\mathsf{id})$.
3. *Set* $\mathbf{u}' = (\mathbf{u}, \mathbf{0}, 0) \in \mathbb{Z}_p^{\ell+Q+1}$ *and compute*

$$\beta_0 := \mathbf{A}^\top \mathbf{s} + \mathbf{e}_0,\ \beta_1 := \mathbf{P}^\top \mathbf{s} + \mathbf{e}_1 + \mathbf{e}_2 + p^{e-1}\mathbf{u}'.$$

4. *Sample* $\mathbf{R} \xleftarrow{\$} \{-1,1\}^{m\times m}$ *and compute*

$$\mathbf{c}_{\mathsf{id}} := [\mathbf{A}|\mathbf{H}_{\mathsf{id}}]^\top \mathbf{s} + \mathbf{R}^\top \mathbf{e}_0.$$

5. *Output the ciphertext* $\mathsf{ct} := (\beta_0, \beta_1, \mathbf{c}_{\mathsf{id}})$.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$ *takes as input* sk *and* ct*,*

1. *Compute* $\mu' = \mathbf{v}'^\top \beta_1 - \mathsf{sk}^\top \begin{pmatrix} \beta_0 \\ \mathbf{c}_{\mathsf{id}} \end{pmatrix} \mod q$.
2. *Output* $\mu \in \mathbb{Z}_p$ *that minimizes* $|p^{e-1} \cdot \mu - \mu'|$.

Parameter Setting. We choose the parameters similarly as in [ACGU20]. To ensure the security and correctness, the parameters should satisfy the following conditions:

1. For the security and correctness of ALS scheme, we set n_{ALS}, m_{ALS}, q_{ALS}, σ_{ALS}, ρ_{ALS}, α_{ALS} as chosen in construction 6.
2. For the security reduction to ALS, we set $n = n_{\text{ALS}}$, $m = m_{\text{ALS}}$, $\sigma = \sigma_{\text{ALS}}$, $\rho = \rho_{\text{ALS}}$.
3. $\mathsf{TrapGen}$ requires $m > 6n \log q$.
4. To the algorithms $\mathsf{SampleLeft}$ and $\mathsf{SampleRight}$, we need $s > m \cdot \omega(\sqrt{\log m})$.
5. We require $\tau > s_1(\mathbf{R}^*\mathbf{J}^*)$ in order to rely on ReRand algorithm for security proof. According to Lemma 5, $s_1(\mathbf{J}^*)$ is bounded by $1/\sqrt{2\pi} \cdot \rho \cdot (\sqrt{\ell + Q + 1} + \sqrt{m} + \sqrt{\lambda})$. Additionally, $s_1(\mathbf{R}^*) < m$, which implies that $s_1(\mathbf{R}^*\mathbf{J}^*) < 1/\sqrt{2\pi} \cdot \rho m \cdot (\sqrt{\ell + Q + 1} + \sqrt{m} + \sqrt{\lambda})$.
6. Leftover hash lemma is carried over from ALS parameters.
7. To ensure the hardness of $\mathsf{LWE}_{q,n,\alpha}$, we require $\alpha q \geq \Omega(n)$.
8. The final magnitude of decryption error must be less than $\frac{1}{2}p^{k-1}$ for the correctness.

Our parameters could be chosen as: $n = \mathsf{poly}(\lambda)$, $m \geq 6n \log q$, $\rho > \omega(\sqrt{\log \lambda})$, $\tau = 1/\sqrt{2\pi} \cdot \rho m \cdot (\sqrt{\ell + Q + 1} + \sqrt{m} + \sqrt{\lambda})$, $\sigma' = 2\sigma\tau$, $q > p^2(\sqrt{\sigma^2 + \sigma'^2} \cdot (\ell + Q + 1) + \rho\sigma \cdot m^{1.5})$.

Lemma 19 (Correctness). *Under the parameters settings described above, the* $\mathsf{IB\text{-}IPFE}$ *scheme in Construction 5 is correct.*

This lemma can be easily verified following the proof of correctness in [ACGU20], with the additional use of the equation $\langle \mathbf{u}', \mathbf{v}' \rangle = \langle (\mathbf{u}, \mathbf{0}, 0), (\mathbf{v}, \boldsymbol{\epsilon}, r) \rangle = \langle \mathbf{u}, \mathbf{v} \rangle$. So we omit the proof details here.

Theorem 6 (sel-SIM Security). *Let 1^λ be the security parameter and consider the ALS* IPFE *scheme (Construction 6) with parameters $q, \sigma, \rho, m, n, \alpha$. As long as ALS is secure under the* $\mathsf{LWE}_{q,n,\alpha}$ *and the ALS parameters satisfy the above parameters choosing conditions, the* $\mathsf{IB\text{-}IPFE}$ *in Construction 5 is* sel*-SIM secure under the* $\mathsf{LWE}_{q,n,\alpha}$ *assumption.*

Proof. We first describe the following simulator.

1. $\mathsf{Setup}^*(1^\lambda, \mathsf{id}^*, \mathsf{st})$: Same as the Setup algorithm in real scheme, except that it additionally samples $r_i \xleftarrow{\$} \mathbb{Z}_p$ for $i \in [Q]$.
2. $\mathsf{KeyGen}^*_{\mathsf{pre}}(1^\lambda, \mathsf{st}, \{r_i\}_{i\in[Q]}, \mathsf{id}, \mathbf{v})$: For key queries that $\mathsf{id} \neq \mathsf{id}^*$, generate the secret keys exactly the same as in real KeyGen algorithm. Otherwise, for key queries $(\mathsf{id}^*, \mathbf{v}_i)$, first set $\mathbf{v}'_i = (\mathbf{v}_i, \boldsymbol{\epsilon}_i, r_i)$ where $\boldsymbol{\epsilon}_i$ is the i-th canonical vector and r_i is the randomness sampled during Setup^* phase. The remaining key components are computed as in real KeyGen.
3. $\mathsf{Enc}^*(1^\lambda, \mathsf{st})$: Let st contains all the queried 1-keys $(\mathsf{id}^*, \mathbf{v}_i^{\mathsf{pre}})_{i\in[Q']}$ and corresponding functional values $d_i^{\mathsf{pre}} = \langle \mathbf{u}^*, \mathbf{v}_i^{\mathsf{pre}} \rangle$. Compute $\widetilde{\mathbf{u}}$ such that $\langle \widetilde{\mathbf{u}}, \mathbf{v}_i^{\mathsf{pre}} \rangle = d_i^{\mathsf{pre}}$ for all $i \in [Q']$. If the adversary has not queried 1-key in pre-challenge phase, the sample a random $\widetilde{\mathbf{u}} \xleftarrow{\$} \mathbb{Z}_p^\ell$. Then generate the challenge ciphertext ct^* by invoking the real Enc on input id^* and $\widetilde{\mathbf{u}}' = (\widetilde{\mathbf{u}}, -r_1, \ldots, -r_Q, 1)$.
4. $\mathsf{KeyGen}^*_{\mathsf{post}}(1^\lambda, \mathsf{st}, \mathsf{id}, \mathbf{v})$: Same as $\mathsf{KeyGen}^*_{\mathsf{pre}}$, except that for the 1-key query $(\mathsf{id}^*, \mathbf{v}_i)$, first compute $\delta_i = d_i^{\mathsf{post}} - \langle \widetilde{\mathbf{u}}, \mathbf{v}_i \rangle$ for $d_i^{\mathsf{post}} = \langle \mathbf{u}^*, \mathbf{v}_i \rangle$ and set $\mathbf{v}'_i = (\mathbf{v}_i, \boldsymbol{\epsilon}_i, r_i + \delta_i)$. Then, compute the remaining key component as in $\mathsf{KeyGen}^*_{\mathsf{pre}}$.

In order to prove the security through a sequence of hybrids, we additionally introduce a set of auxiliary algorithms.

$\mathsf{Setup}^*_1(1^\lambda, \mathsf{id}^*, \mathsf{st})$

1. Sample $(\mathbf{A}, \mathbf{T}_\mathbf{A}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$.
2. Set $\mathbf{B} := \mathbf{A} \cdot \mathbf{R}^* - \mathsf{H}(\mathsf{id}^*)\mathbf{G}$ for $\mathbf{R}^* \xleftarrow{\$} \{-1, 1\}^{m\times m}$.
3. Sample $\mathbf{J}^* \leftarrow \mathcal{D}_{\mathbb{Z}^{m\times(\ell+Q+1)},\rho}$ and set $\mathbf{P} := [\mathbf{A}|\mathbf{H}(\mathsf{id}^*)] \cdot \mathbf{J}^* \mod q$.
4. Sample $r_i \xleftarrow{\$} \mathbb{Z}_p$ for $i \in [Q]$.

5. Output $\mathsf{mpk} := (\mathbf{A}, \mathbf{B}, \mathbf{P}), \mathsf{msk} := (\mathbf{T}_{\mathbf{A}}, \mathbf{J}^*, \{r_i\}_{i\in[Q]})$.

$\mathsf{KeyGen}_1^*(1^\lambda, \mathsf{st}, \{r_i\}_{i\in[Q]}, \mathsf{id}, \mathbf{v})$

– If $\mathsf{id} \neq \mathsf{id}^*$, generate the secret key sk as in real KeyGen except for the matrix $\mathbf{K}_{\mathsf{id}}$ being sampled using $\mathbf{T}_{\mathbf{G}}$ instead of $\mathbf{T}_{\mathbf{G}}$.

$$\mathbf{K}_{\mathsf{id}} \leftarrow \mathsf{SampleRight}(\mathbf{A}, \mathbf{R}^*, \mathsf{H}(\mathsf{id}) - \mathsf{H}(\mathsf{id}^*), \mathbf{P}, s).$$

– If $\mathsf{id} = \mathsf{id}^*$, $\mathbf{K}_{\mathsf{id}^*}$ is set as $\mathbf{J}^*$ satisfying $[\mathbf{A}|\mathbf{H}(\mathsf{id}^*)] \cdot \mathbf{K}_{\mathsf{id}^*} = \mathbf{P}$. For pre-challenge key queries, set $\mathbf{v}_i' = (\mathbf{v}_i, \boldsymbol{\epsilon}_i, r_i)$, where r_i is chosen in Setup_1^*. For post-challenge key queries, first compute a dummy vector $\widetilde{\mathbf{u}}$ satisfying $\langle \widetilde{\mathbf{u}}, \cdot \rangle$ stays consistent with all ever queried pre-challenge keys and the challenge message. Then, compute $\delta_i = \langle \mathbf{u}^*, \mathbf{v}_i \rangle - \langle \widetilde{\mathbf{u}}, \mathbf{v}_i \rangle$ and set $\mathbf{v}_i' = (\mathbf{v}_i, \boldsymbol{\epsilon}_i, \delta_i + r_i)$. Return $\mathsf{sk}_{\mathsf{id}^*,i} := (\mathbf{v}_i', \mathbf{K}_{\mathsf{id}^*} \cdot \mathbf{v}_i')$.

$\mathsf{Enc}_1^*(1^\lambda, \mathsf{st})$: Same as real Enc, except that $\mathbf{u}'$ is set as $(\widetilde{\mathbf{u}}, -r_1, \ldots, -r_Q, 1)$, where $\widetilde{\mathbf{u}}$ is the dummy vector computed to satisfy consistent constraints and $\{r_i\}_{i\in[Q]}$ are chosen is Setup_1^*.

Now we describe the following hybrids to prove the security.

Hybrid $\mathcal{H}_0$: Real experiment.

Hybrid $\mathcal{H}_1$: Switch the real Setup to Setup_1^*. The main differences lie in the generation of $(\mathbf{B}, \mathbf{P})$ and the sampling of $\{r_i\}_{i\in[Q]}$, which will be later used for answering 1-key. We have $\mathcal{H}_0 \approx_s \mathcal{H}_1$ by applying Leftover hash lemma (Lemma 6).

Hybrid $\mathcal{H}_2$: Switch the real KeyGen to KeyGen_1^*. For the key queries that $\mathsf{id} \neq \mathsf{id}^*$, the key matrices $\mathbf{K}_{\mathsf{id}}$ are generated using $\mathsf{SampleLeft}$ in $\mathcal{H}_1$ whereas in $\mathcal{H}_2$, these are computed by $\mathsf{SampleRight}$. By employing the Lemma 3 and 4, the resulting distributions are statistically indistinguishable. For the remaining queries on $\mathsf{id} = \mathsf{id}^*$, we can claim similarly as $\mathbf{K}_{\mathsf{id}^*}$ turns to be generated using 2-stage sampling in $\mathcal{H}_2$. Therefore, the distribution of $\mathbf{K}_{\mathsf{id}^*}$ is statistically close in $\mathcal{H}_1$ and $\mathcal{H}_2$ by Lemma 7. Notice that r_i is chosen uniformly in $\mathbb{Z}_p$ during the Setup_1^* phase. Hence, the two distributions $\{r_i\}$ and $\{\delta_i + r_i\}$ are statistically indistinguishable for post-challenge queries in two hybirds.

Hybrid $\mathcal{H}_3$: Switch to choosing $\mathbf{A}$ uniformly from $\mathbb{Z}_q^{n\times m}$. $\mathcal{H}_2$ and $\mathcal{H}_3$ are statistically indistinguishable by the property of $\mathsf{TrapGen}$ algorithm.

Hybrid $\mathcal{H}_4$: Switch the real Enc to Enc_1^*. We reduce the distinguishing advantage of $\mathcal{H}_4$ and $\mathcal{H}_3$ to the security of ALS IPFE scheme.

On receiving the public key $(\mathbf{A}_{\mathrm{ALS}}, \mathbf{D}_{\mathrm{ALS}}) \in \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^{(\ell+Q+1)\times n}$ from the ALS challenger (as described in Construction 6), we simulate the view of the distinguisher for $\mathcal{H}_3$ versus $\mathcal{H}_4$ as follows.

Setup: Set $\mathbf{A} := \mathbf{A}_{\mathrm{ALS}}^\top$. Sample $\mathbf{R}^*$ and $\mathbf{J}^*$, compute $\mathbf{B}$ as in Setup_1^* and set $\mathbf{P} := \mathbf{D}_{\mathrm{ALS}}^\top + \mathbf{A}\mathbf{R}^*\mathbf{J}^*$. Return the $\mathsf{mpk} = (\mathbf{A}, \mathbf{B}, \mathbf{P})$ to the distinguisher.

KeyGen: For key query $(\mathsf{id}, \mathbf{v})$,

- For $\mathsf{id} \neq \mathsf{id}^*$, generate the secret keys as in $\mathcal{H}_3$.
- For $(\mathsf{id}^*, \mathbf{v}_i)$, set $\mathbf{v}_i'$ as $(\mathbf{v}_i, \boldsymbol{\epsilon}_i, r_i)$ for pre-challenge queries and as $(\mathbf{v}_i, \boldsymbol{\epsilon}_i, \delta_i + r_i)$ for post-challenge queries, where δ_i are computed as in KeyGen_1^*. Send the key query $\mathbf{v}_i'$ to the ALS challenger, and obtain $\mathsf{sk}_{\mathbf{v}_i}$. Compute and return $\mathsf{sk}_{\mathbf{v}_i'} = \begin{bmatrix} \mathsf{sk}_{\mathbf{v}_i'} \\ \mathbf{J}^* \cdot \mathbf{v}_i' \end{bmatrix}$.

Enc: Sample $\widetilde{\mathbf{u}} \xleftarrow{\$} \mathbb{Z}_p^\ell$ if there is no pre-challenge key queries for id^*. Otherwise, compute $\widetilde{\mathbf{u}}$ such that $\langle \widetilde{\mathbf{u}}, \mathbf{v}_i^{\mathsf{pre}} \rangle = \langle \mathbf{u}^*, \mathbf{v}_i^{\mathsf{pre}} \rangle$ for all key queries $(\mathsf{id}^*, \cdot)$ in the pre-challenge phase. Set

$$\mathbf{u}_0' = (\mathbf{u}^*, \mathbf{0}, 0),\ \mathbf{u}_1' = (\widetilde{\mathbf{u}}, -r_1, \ldots, -r_Q, 1)$$

and send $(\mathbf{u}_0', \mathbf{u}_1')$ to the ALS challenger. Upon receiving $(\mathsf{ct}_0, \mathsf{ct}_1)$, compute and return

$$\beta_0 := \mathsf{ct}_0,\ \mathbf{c}_{\mathsf{id}^*} := (\mathbf{R}^*)^\top \beta_0,$$
$$\beta_1 := \mathsf{ct}_1 + \mathsf{ReRand}(\mathbf{R}^*\mathbf{J}^*, \mathsf{ct}_0, \sigma, \tau).$$

To analyze the above reduction, first notice that all the key queries and challenge message vector submitted to the ALS challenger are admissible according to the computation principle of dummy vector $\widetilde{\mathbf{u}}$. In other words, it holds that $\langle \mathbf{u}'_0, \mathbf{v}'_i \rangle = \langle \mathbf{u}'_1, \mathbf{v}'_i \rangle$ for all $i \in [Q]$. For $\mathsf{ct}_0 = \mathbf{A}_{\mathrm{ALS}} \cdot \mathbf{s} + \mathbf{e}_0$, we know that $\mathsf{ReRand}(\mathbf{J}^*, \mathsf{ct}_0, \sigma, \tau) = (\mathbf{A}\mathbf{J}^*)^\top \cdot \mathbf{s} + \mathbf{e}'$ for $\tau > s_1(\mathbf{R}^*\mathbf{J}^*)$, where $\mathbf{e}' \approx_s \mathcal{D}_{\mathbb{Z}^{\ell+Q+1}, 2\sigma\tau}$ by the property of ReRand (Lemma 10). Hence, we have

$$\begin{aligned}\beta_1 &= \mathbf{D}_{\mathrm{ALS}} \cdot \mathbf{s} + \mathbf{e}_1 + p^{e-1} \cdot \mathbf{u}'_b + (\mathbf{A}\mathbf{J}^*)^\top \cdot \mathbf{s} + \mathbf{e}' \\ &= \mathbf{P}^\top \cdot \mathbf{s} + \mathbf{e}_1 + \mathbf{e}' + p^{e-1} \cdot \mathbf{u}'_b,\end{aligned}$$

where $\mathbf{e}_1$ is distributed as $\mathcal{D}_{\mathbb{Z}^{\ell+Q+1}, \sigma}$. Thus, the noise in β_1 is statistically close to $(\mathbf{e}_1 + \mathbf{e}_2)$ obtained in the real Enc algorithm ensured by our parameters setting.

Therefore, we perfectly simulate $\mathcal{H}_3$ or $\mathcal{H}_4$ depending on the challenge bit chosen by the ALS challenger.

Hybrid $\mathcal{H}_5$: Switch to the ideal experiment conducted by the simulator $(\mathsf{Setup}^*, \mathsf{KeyGen}^*_{\mathsf{pre}}, \mathsf{Enc}^*, \mathsf{KeyGen}^*_{\mathsf{post}})$. $\mathcal{H}_4 \approx_s \mathcal{H}_5$ follows by the indistinguishability of $\mathcal{H}_0$ and $\mathcal{H}_3$.

□

A.2 The ALS Inner-Product Functional Encryption Scheme

We recall the variant of ALS IPFE scheme originally introduced in [ALS16] and later simplified in [ACGU20]. The construction in [ACGU20] utilizes a different master secret key $\mathbf{Z} \leftarrow \mathcal{D}_{\mathbb{Z}^{\ell \times m}, \rho}$, sampled from a significantly simpler distribution compared to the original scheme. Additionally, in their work, the security proof leverages the rerandomization technique [KY16] inspired by [WFL19]. In our construction, we consider a slightly different function class, focusing on inner products modulo a prime p.

Construction 6 (FE for inner products modulo a prime p).

$\mathsf{Setup}(1^\lambda, p, \ell)$ *takes as input the security parameter* 1^λ*, modulus prime* p *and the length of message and key vector* ℓ*,*

1. *Set parameters* $n, m, \sigma, \rho, q = p^k$ *for some integer* k*.*
2. *Sample* $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$ *and* $\mathbf{Z} \leftarrow \mathcal{D}_{\mathbb{Z}^{\ell \times m}, \rho}$*.*
3. *Compute* $\mathbf{D} = \mathbf{Z} \cdot \mathbf{A} \in \mathbb{Z}_q^{\ell \times n}$*.*
4. *Output the public and master secret keys*

$$\mathsf{mpk} \coloneqq (\mathbf{A}, \mathbf{D}), \mathsf{msk} \coloneqq \mathbf{Z}.$$

$\mathsf{KeyGen}(\mathsf{msk}, \mathbf{y}, \mathsf{st})$ *takes as input* msk*, key vector* $\mathbf{y}$ *and the state value* st*,*

1. *Check the state* $\mathsf{st} = (\mathbf{y}_i, \overline{\mathbf{y}}_i, \mathbf{z}_i)$*, where the* $\mathbf{y}_i$*'s are a subset of the key queries that have been made so far, and the* $(\overline{\mathbf{y}}_i, \mathbf{z}_i)$*'s are the corresponding secret keys.*
   - *If* $\mathbf{y}$ *is linearly independent from ever queried key vectors modulo* p*, set* $\overline{\mathbf{y}} = \mathbf{y} \in \mathbb{Z}_p^\ell$*,* $\mathbf{z}_{\mathbf{y}} = \mathbf{Z}^\top \overline{\mathbf{y}}$ *and add* $(\mathbf{y}, \overline{\mathbf{y}}, \mathbf{z})$ *to* st*.*
   - *If* $\mathbf{y}$ *can be linearly described by* $\mathbf{y} = \sum_i d_i \mathbf{y}_i$ *for some* $d_i \in \mathbb{Z}_p$*, then set* $\overline{\mathbf{y}} = \sum_i d_i \overline{\mathbf{y}}_i$ *and* $\mathbf{z}_{\mathbf{y}} = \sum_i d_i \overline{\mathbf{z}}_i$*.*
2. *Output* $(\overline{\mathbf{y}}, \mathbf{z}_{\mathbf{y}})$*.*

$\mathsf{Enc}(\mathsf{mpk}, \mathbf{x})$ *takes as input* mpk *and a message* $\mathbf{x}$*,*

1. *Sample* $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$*,* $\mathbf{e}_0 \leftarrow \mathcal{D}_{\mathbb{Z}^m, \sigma}$*,* $\mathbf{e}_1 \leftarrow \mathcal{D}_{\mathbb{Z}^\ell, \sigma}$*.*
2. *Compute*

$$\mathsf{ct}_0 \coloneqq \mathbf{A} \cdot \mathbf{s} + \mathbf{e}_0, \ \mathsf{ct}_1 \coloneqq \mathbf{D} \cdot \mathbf{s} + \mathbf{e}_1 + p^{k-1} \cdot \mathbf{x}.$$

3. *Output the ciphertext* $\mathsf{ct} \coloneqq (\mathsf{ct}_0, \mathsf{ct}_1)$*.*

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$ *takes as input* sk *and* ct,

1. *Compute* $\mu' = \overline{\mathbf{y}}^\top \mathsf{ct}_1 - \mathbf{z}_{\mathbf{y}}^\top \mathsf{ct}_0 \mod q$.
2. *Output* $\mu \in \mathbb{Z}_p$ *that minimizes* $|p^{e-1} \cdot \mu - \mu'|$.

Parameters Setting. The parameters should satisfy the following constraints aiming for the correctness and security.

1. The final magnitude of decryption error must be less than $\frac{1}{2}p^{k-1}$ for the correctness.
2. To ensure the hardness of $\mathsf{LWE}_{q,n,\alpha}$, we require $\alpha q \geq \Omega(n)$.
3. We require $\tau > s_1(\mathbf{Z})$ in order to rely on ReRand algorithm for security proof. According to Lemma 5, $s_1(\mathbf{Z})$ is bounded by $1/\sqrt{2\pi} \cdot \rho \cdot (\sqrt{\ell} + \sqrt{m} + \sqrt{\lambda})$.
4. To ensure large enough entropy, we require $\rho > \omega(\sqrt{\log \lambda})$ and $m \geq (2n \log q + 2n)/\log(4/3)$.

Our parameters could be chosen as: $n = \mathsf{poly}(\lambda)$, $m = 2(n \log q)$, $\rho > \omega(\sqrt{\log \lambda})$, $\tau = 1/\sqrt{2\pi} \cdot \rho \cdot (\sqrt{\ell} + \sqrt{m} + \sqrt{\lambda})$, $\sigma = 2\alpha q \tau$, $q > 2p^2\sigma(\ell + \rho\sqrt{m})$.

Lemma 20 (Correctness). *Under the parameters set as described above, the* IPFE *scheme in construction 6 is correct.*

Proof (Sketch). This lemma can be easily verified following the proof of correctness in [ALS16, ACGU20], with a slightly difference analyze for decryption noise as follows. After the first step in decryption, we have $\mu' = e_{\mathsf{dec}} + p^{k-1} \cdot \langle \overline{\mathbf{y}}, \mathbf{x} \rangle$, where $e_{\mathsf{dec}} = \overline{\mathbf{y}}^\top \mathbf{e}_1 - \overline{\mathbf{y}}^\top \cdot \mathbf{Z} \cdot \mathbf{e}_0$. We can bound this error term by $p \cdot \sigma\ell + p \cdot \rho\sigma\sqrt{m} \leq 2p\sigma(\ell + \rho\sqrt{m}) \leq \frac{1}{2}p^{k-1}$, which is ensured by the choice of q. □

Theorem 7 ([ALS16, WFL19, ACGU20]). *According to the above parameters setting, the* IPFE *scheme in Construction 6 is secure, assuming that* $\mathsf{LWE}_{q,n,\alpha}$ *is hard.*

The security of Construction 6 follows from the previous security of schemes [ALS16, WFL19, ACGU20]. Hence, we omit the proof details here.

A.3 Inner-Product Functional Encryption Scheme Supporting Multiple Challenges

We now propose an IPFE scheme with simulation-based security allowing a single key query and a bounded number of encryption queries, designed to facilitate the generic construction of updatable Predicate encryption in construction 2. The IPFE construction in [ALMT20] provided a generic method of achieving SIM-based IPFE by relying on an IND-based IPFE scheme with extended programmable slots in both the message and key space. Our setting differs that we handle multiple challenges with a single secret key, whereas their scheme allows the adversary to ask for multiple linearly independent keys but only a single challenge. The key to achieving simulation security in an IPFE scheme lies in simulating the challenge ciphertext based on the functional value, rather than the challenge message itself. The additional slots enables us to fix the trade-off and correct the decryption result during the security proof in an indistinguishable manner. Fortunately, we observe that this approach applies similarly to our setting.

To encode Q_c inner products in both the key and challenge ciphertexts, we extend the underlying dimension by adding $2Q_c$ additional slots, where each pair of slots encodes one inner product. In the scheme, each message vector $\mathbf{x} \in \mathbb{Z}_p^\ell$ is extended to $(\mathbf{x}, \mathbf{0}, \mathbf{0}) \in \mathbb{Z}_p^{\ell+2Q_c}$, and each key vector $\mathbf{y}$ is encoded as $\mathbf{y}' = (\mathbf{y}, \mathbf{1}, \mathbf{r})$. In the proof, however, we modify $\mathbf{x}_j$ to $(\widetilde{\mathbf{x}}_j, -r_j \cdot \boldsymbol{\epsilon}_j, \boldsymbol{\epsilon}_j)$ and $\mathbf{y}$ to $(\mathbf{y}, \mathbf{1}, \mathbf{r}')$, which results in a decryption output $\langle \widetilde{\mathbf{x}}_j, \mathbf{y} \rangle - r_j + r'_j$. We can then compute $\mathbf{x}'_j$ without needing $\mathbf{x}_j$ by either sampling $\widetilde{\mathbf{x}}_j$ uniformly or by computing it under specific conditions and adjusting r'_j to achieve the correct decryption value.

Construction 7 (Q_c-challenges SIM-secure IPFE). *Our construction uses an IPFE scheme* $\mathsf{iFE} = (\mathsf{iSetup}, \mathsf{iEnc}, \mathsf{iKeyGen}, \mathsf{iDec})$ *as the main building block.*

$\mathsf{Setup}(1^\lambda, p, \ell)$ *takes as input the security parameter* 1^λ*, modulus prime* p *and the length of message and key vector* ℓ,

1. *Set* $L = \ell + 2Q_c$, *and run* $(\mathsf{iMPK}, \mathsf{iMSK}) \leftarrow \mathsf{iSetup}(1^\lambda, 1^L, p)$.
2. *Output the public and master secret keys*

$$\mathsf{mpk} := \mathsf{iMPK}, \mathsf{msk} := \mathsf{iMPK}.$$

$\mathsf{Enc}(\mathsf{mpk}, \mathbf{x})$ *takes as input* mpk *and a message* $\mathbf{x}$,

1. *Set* $\mathbf{x}' = (\mathbf{x}, \mathbf{0}, \mathbf{0}) \in \mathbb{Z}_p^L$.
2. *Compute and return* $\mathsf{ct} := \mathsf{iCT} \leftarrow \mathsf{iEnc}(\mathsf{iMPK}, \mathbf{x}')$.

$\mathsf{KeyGen}(\mathsf{msk}, \mathbf{y})$ *takes as input* msk *and key vector* $\mathbf{y}$,

1. *Sample* $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_p^{Q_c}$ *and set* $\mathbf{y}' = (\mathbf{y}, \mathbf{1}, \mathbf{r}) \in \mathbb{Z}_p^L$.
2. *Compute and return* $\mathsf{sk}_\mathbf{y} := \mathsf{iSK}_{\mathbf{y}'} \leftarrow \mathsf{iKeyGen}(\mathsf{iMSK}, \mathbf{y}')$.

$\mathsf{Dec}(\mathsf{sk}_\mathbf{y}, \mathsf{ct})$ *takes as input* sk *and* ct,

1. *Parse the secret key as* $\mathsf{sk}_\mathbf{y} = \mathsf{iSK}_{\mathbf{y}'}$ *and the ciphertext as* $\mathsf{ct} = \mathsf{iCT}$.
2. *Compute and output*

$$\mu \leftarrow \mathsf{iDec}(\mathsf{iSK}_{\mathbf{y}'}, \mathsf{iCT}).$$

Lemma 21 (Correctness). *If the underlying IPFE scheme* iFE *is correct, then the* IPFE *scheme in Construction 7 is correct.*

Proof (Sketch). The correctness of Construction 7 follows from the correctness of underlying iFE scheme and the relationship that $\langle (\mathbf{x}, \mathbf{0}, \mathbf{0}), (\mathbf{y}, \mathbf{1}, \mathbf{r}) \rangle = \langle \mathbf{x}, \mathbf{y} \rangle$. □

Theorem 8 ([ALMT20]). *Assume that the underlying IPFE scheme* iFE *is adaptively IND-based secure against single key query and at most* Q_c *challenges, then the IPFE scheme (Construction 7) achieves adaptive simulation-based security against single key query and at most* Q_c *challenges.*

Proof. We prove the security by first defining the following simulator and then analyzing the differences between the real experiment and the ideal experiment conducted by the simulator through a sequence of hybrids.

1. $\mathsf{Setup}^*(1^\lambda, \mathsf{st})$: Same as the Setup algorithm in real scheme, except that the simulator additionally samples $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_p^{Q_c}$.
2. $\mathsf{Enc}^*_{\mathsf{pre}}(1^\lambda, \mathsf{st})$: For the encryption queries that are submitted before the single key query, the simulator first samples $\widetilde{\mathbf{x}}_j \xleftarrow{\$} \mathbb{Z}_p^\ell$ and then sets

$$\mathbf{x}'_j = (\widetilde{\mathbf{x}}_j, -r_j \cdot \boldsymbol{\epsilon}_j, \boldsymbol{\epsilon}_j) \in \mathbb{Z}_p^L,$$

   where $\boldsymbol{\epsilon}_j \in \{0,1\}^{Q_c}$ is the j-th canonical vector. Compute and return $\mathsf{ct}^*_j \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathbf{x}'_j)$.
3. $\mathsf{KeyGen}^*(1^\lambda, \mathsf{st}, \mathsf{info}(\{\mathbf{x}^*_j\}), \mathbf{y})$: Let $\mathsf{info}(\{\mathbf{x}^*_j\})$ contain all the inner products $d^{\mathsf{pre}}_j = \langle \mathbf{x}^*_j, \mathbf{y} \rangle$ that the adversary could learn from $Q' \le Q_c$ encryption queries and single key query. For $j \in [Q']$, the simulator computes $\delta_j = d^{\mathsf{pre}}_j - \langle \widetilde{\mathbf{x}}_j, \mathbf{y} \rangle$ and sets $\mathbf{y}' = (\mathbf{y}, \mathbf{1}, \mathbf{r}') \in \mathbb{Z}_p^L$, where $\mathbf{r}'$ is defined as

$$r'_j = \begin{cases} r_j + \delta_j \mod p & \text{for } j \in [Q'] \\ r_j & \text{for } j \in [Q'+1; Q_c] \end{cases}$$

   Compute and return $\mathsf{sk}_\mathbf{y} = \mathsf{iSK}_{\mathbf{y}'} \leftarrow \mathsf{iKeyGen}(\mathsf{iMSK}, \mathbf{y}')$.
4. $\mathsf{Enc}^*_{\mathsf{post}}(1^\lambda, \mathsf{info}(\{\mathbf{x}^*_j\}), \mathsf{st})$: For each encryption query, on giving the corresponding functional value d^{post}_j, the simulator first computes $\widetilde{\mathbf{x}}_j$ to satisfy

$$\langle \widetilde{\mathbf{x}}_j, \mathbf{y} \rangle = d^{\mathsf{post}}_j \mod p.$$

   Then set $\mathbf{x}'_j = (\widetilde{\mathbf{x}}_j, -r_j \cdot \boldsymbol{\epsilon}_j, \boldsymbol{\epsilon}_j) \in \mathbb{Z}_p^L$, compute and return $\mathsf{ct}^*_j \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathbf{x}'_j)$.

Now we describe the following hybrids to prove the security.

Hybrid $\mathcal{H}_0$: Real experiment.

Hybrid $\mathcal{H}_1$: Modify the Setup algorithm to additionally pre-sample $r_j \xleftarrow{\$} \mathbb{Z}_p$ for $j \in [Q_c]$, which will be used to simulate single-key and multi-challenge ciphertexts. There is only conceptual difference, thus we have $\mathcal{H}_0 \equiv \mathcal{H}_1$.

Hybrid $\mathcal{H}_2$: The challenger generates the single queried key as in the real KeyGen procedure, except for how the randomness encoded in vector $\mathbf{y}'$ is computed. In more details, if there exists $Q' \leq Q_c$ encryption queries before the single key query, then the challenger first samples $\{\widetilde{\mathbf{x}}_j \xleftarrow{\$} \mathbb{Z}_p^\ell\}_{j\in[Q']}$ and computes

$$\delta_j \coloneqq d_j^{\mathsf{pre}} - \langle \widetilde{\mathbf{x}}_j, \mathbf{y} \rangle \mod p,$$

where d_j^{pre} is the decryption results of the single key query and all ever queried ciphertexts. Then, set $\mathbf{y}'$ as $(\mathbf{y}, \mathbf{1}, \mathbf{r}')$ for $\mathbf{r}'$ defined as

$$r'_j = \begin{cases} r_j + \delta_j \mod p & \text{for } j \in [Q'] \\ r_j & \text{for } j \in [Q'+1; Q_c] \end{cases}$$

If there is no encryption queries before, then set $\mathbf{r}' \coloneqq \mathbf{r}$.

We notice that the only difference between $\mathcal{H}_1$ and $\mathcal{H}_2$ is the chosen randomness $(r'_1, \ldots, r'_{Q'})$ in the key vector in the case that the adversary have queried challenge ciphertexts before the current key query. In $\mathcal{H}_1$, these components are directly chosen as uniformly random from $\mathbb{Z}_p$, while in $\mathcal{H}_2$, these are set as $r_j + \delta_j$. As each r_j is sampled uniformly for $j \in [Q']$, the resulting $r_j + \delta_j$ is still distributed as uniformly random in $\mathbb{Z}_p$. Hence, the distributions $\{r_j\}_{j\in[Q']}$ and $\{r_j + \delta_j\}_{j\in[Q']}$ in the two hybrids are identical distributed for $r_j \xleftarrow{\$} \mathbb{Z}_p$ and $\delta_j \in \mathbb{Z}_p$. Moreover, as the last $2Q_c$ coordinates of each message vector equal zero, the output of the decryption remains consistent. Therefore, we have $\mathcal{H}_1 \equiv \mathcal{H}_2$.

Hybrid $\mathcal{H}_3$: The challenger modifies the method for extending message vectors. Each extended vector is set as $\mathbf{x}'_j = (\widetilde{\mathbf{x}}_j, -r_j \cdot \boldsymbol{\epsilon}_j, \boldsymbol{\epsilon}_j) \in \mathbb{Z}_p^L$, where the choice of $\widetilde{\mathbf{x}}_j$ varies depending on the timing of the encryption query.

$$\widetilde{\mathbf{x}}_j = \begin{cases} \widetilde{\mathbf{x}}_j \xleftarrow{\$} \mathbb{Z}_p^\ell & \text{for } j \in [Q'] \\ \widetilde{\mathbf{x}}_j \in \mathbb{Z}_p^\ell \text{ s.t. } \langle \widetilde{\mathbf{x}}_j, \mathbf{y} \rangle = d_j^{\mathsf{post}} \mod p & \text{for } j \in [Q'+1; Q_c] \end{cases}$$

The differences of $\mathcal{H}_2$ and $\mathcal{H}_3$, which is equivalent to the ideal experiment, lies in how the message vectors are extended for each encryption query. Next, we demonstrate the indistinguishability of $\mathcal{H}_2$ and $\mathcal{H}_3$ by constructing an adversary $\mathcal{B}$ against the full-fledged security of the underlying iFE scheme as follows.

Setup: Upon receiving $(1^\lambda, p, \ell)$ from the distinguisher, $\mathcal{B}$ submits $(1^\lambda, p, L)$ to its iFE challenger and obtains iMPK. It sets $\mathsf{mpk} \coloneqq \mathsf{iMPK}$ and samples $\mathbf{v} \xleftarrow{\$} \mathbb{Z}_p^{Q_c}$. Then, it returns mpk to the distinguisher.

Enc: For encryption queries that submitted before the single key query, $\mathcal{B}$ computes

$$\mathbf{x}'_{j,0} \coloneqq (\mathbf{x}_j^*, \mathbf{0}, \mathbf{0}),\ \mathbf{x}'_{j,1} \coloneqq (\widetilde{\mathbf{x}}_j, -r_j \cdot \mathbf{e}_j, \mathbf{e}_j),$$

for $\widetilde{\mathbf{x}}_j \xleftarrow{\$} \mathbb{Z}_p^\ell$ and sends the challenge query $(\mathbf{x}'_{j,0}, \mathbf{x}'_{j,1})$ to the iFE challenger. Once it obtains $\mathsf{iCT}_{j,b}$, sets $\mathsf{ct}_j^* \coloneqq \mathsf{iCT}_{j,b}$ as the return to the distinguisher.

KeyGen: On receiving the key query $\mathbf{y}$, $\mathcal{B}$ extends it to $\mathbf{y}' = (\mathbf{y}, \mathbf{1}, \mathbf{r}')$ as in KeyGen^* algorithm, where r'_j is set as $r_j + \delta_j$ for $j \in [Q']$ and as r_j for the remaining $Q_c - Q'$ coordinates. It then queries the iFE challenger with $\mathbf{y}'$ and gets secret key $\mathsf{iSK}_{\mathbf{y}'}$. Then, $\mathcal{B}$ sets $\mathsf{sk}_{\mathbf{y}} \coloneqq \mathsf{iSK}_{\mathbf{y}'}$ and returns it.

Enc: For encryption queries that submitted after the single key query, $\mathcal{B}$ acts almost identically as dealing with encryption queries before. Namely, it computes

$$\mathbf{x}'_{j,0} \coloneqq (\mathbf{x}_j^*, \mathbf{0}, \mathbf{0}),\ \mathbf{x}'_{j,1} \coloneqq (\widetilde{\mathbf{x}}_j, -r_j \cdot \boldsymbol{\epsilon}_j, \boldsymbol{\epsilon}_j).$$

However, the vector $\widetilde{\mathbf{x}}_j$ is computed to satisfy $\langle \widetilde{\mathbf{x}}_j, \mathbf{y} \rangle = d_j^{\mathsf{post}} \mod p$. As before, $\mathcal{B}$ queries the iFE challenger with the challenge pair $(\mathbf{x}'_{j,0}, \mathbf{x}'_{j,1})$ and forwards the received iCT to the distinguisher.

We claim that all queries to the iFE challenger are admissible. For the encryption queries with index $j \in [Q']$, we have

$$\begin{aligned}\langle \mathbf{x}'_{j,0}, \mathbf{y}' \rangle &= \langle \mathbf{x}^*_j, \mathbf{y} \rangle + \langle \mathbf{0}, \mathbf{1} \rangle + \langle \mathbf{0}, \mathbf{r}' \rangle = d_j^{\mathsf{pre}} \mod p,\\ \langle \mathbf{x}'_{j,1}, \mathbf{y}' \rangle &= \langle \widetilde{\mathbf{x}}_j, \mathbf{y} \rangle + \langle -r_j \cdot \boldsymbol{\epsilon}_j, \mathbf{1} \rangle + \langle \boldsymbol{\epsilon}_j, \mathbf{r}' \rangle\\ &= \langle \widetilde{\mathbf{x}}_j, \mathbf{y} \rangle - r_j + r_j + \delta_j = d_j^{\mathsf{pre}} \mod p,\end{aligned}$$

which follows from that $\delta_j = d_j^{\mathsf{pre}} - \langle \widetilde{\mathbf{x}}_j, \mathbf{y} \rangle$. Similarly, for the encryption queries after the single key query, it hold that

$$\begin{aligned}\langle \mathbf{x}'_{j,0}, \mathbf{y}' \rangle &= \langle \mathbf{x}^*_j, \mathbf{y} \rangle + \langle \mathbf{0}, \mathbf{1} \rangle + \langle \mathbf{0}, \mathbf{r}' \rangle = d_j^{\mathsf{post}} \mod p,\\ \langle \mathbf{x}'_{j,1}, \mathbf{y}' \rangle &= \langle \widetilde{\mathbf{x}}_j, \mathbf{y} \rangle + \langle -r_j \cdot \boldsymbol{\epsilon}_j, \mathbf{1} \rangle + \langle \boldsymbol{\epsilon}_j, \mathbf{r}' \rangle\\ &= \langle \widetilde{\mathbf{x}}_j, \mathbf{y} \rangle - r_j + r_j = d_j^{\mathsf{post}} \mod p.\end{aligned}$$

Since the constructed algorithm $\mathcal{B}$ is admissible in the iFE security game, it perfectly simulate the view of either $\mathcal{H}_2$ or $\mathcal{H}_3$, depending on the challenge bit chosen by the iFE challenger. Therefore, the advantage of distinguishing $\mathcal{H}_2$ from $\mathcal{H}_3$ is bounded by the adversary's advantage against the underlying iFE scheme.

□

B ABE Schemes with Linearly Constructible Ciphertexts

We present two ABE schemes having linearly constructible ciphertexts (LCC). Thus, both can be transformed into single-hop updatable ABE schemes using Construction 1. As the inputs of the IB-IPFE scheme need to be in $\mathbb{Z}_q$, the errors are sampled from a truncated Gaussian distribution. By Lemma 11, for the right choice of parameters, both schemes remain secure.

B.1 KP-ABE scheme with LCC

We show that the KP-ABE scheme by Boneh et al. [BGG+14] with truncated errors has linear constructible ciphertexts. Since the recently constructed RAM-ABE of Dong et al. [DHM+24] has the exact same structure for ciphertexts as the KP-ABE scheme of [BGG+14], it means we can even obtain CU-ABE for RAM programs.

Construction 8 ([BGG+14]).

$\mathsf{Setup}\ (1^\lambda, 1^\ell, 1^d)$**:** *Sample* $\mathbf{A}_i \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$ *for all* $i \in [\ell]$, $\mathbf{w} \xleftarrow{\$} \mathbb{Z}_q^n$ *and* $(\mathbf{A}, \mathbf{T}_\mathbf{A}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$. *Output*

$$\mathsf{mpk} = (\mathbf{A}, \{\mathbf{A}_i\}_{i\in\ell}, \mathbf{w}).$$

$\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}, \mu \in \{0,1\})$**:** *Sample* $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\mathbf{e}_0 \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma,\le q}$, $\mathbf{e}_1 \leftarrow \mathcal{D}_{\mathbb{Z}^{m\ell},\sigma,\le q}$ *and* $e' \leftarrow \mathcal{D}_{\mathbb{Z},\sigma,\le q}$. *Let* $\mathbf{H} = [\mathbf{A}_1||\dots||\mathbf{A}_\ell] + \mathbf{x}^\top \otimes \mathbf{G}$. *Return*

$$\mathsf{ct} = (\beta_0 = \mathbf{w}^\top\mathbf{s} + e' + \lceil q/2\rceil\mu, \beta_1^\top = \mathbf{s}^\top\mathbf{A} + \mathbf{e}_0^\top, \beta_2^\top = \mathbf{s}^\top\mathbf{H} + \mathbf{e}_1^\top, \mathbf{x}).$$

$\mathsf{KeyGen}(\mathsf{msk}, f)$**:** *Run* $\mathbf{H}_f := \mathsf{EvalF}(f, \{\mathbf{A}_i\}_{i\in[n]})$ *and* $\mathbf{A}_f := [\mathbf{A}_1||\dots||\mathbf{A}_\ell]\mathbf{H}_f$. *Set*

$$\mathbf{k}_f \leftarrow \mathsf{SampleLeft}(\mathbf{A}, \mathbf{A}_f, \mathbf{T}_\mathbf{A}, \mathbf{w}, \rho)$$

such that $[\mathbf{A}||\mathbf{A}_f]\mathbf{k}_f = \mathbf{w}$. *Return* $\mathsf{sk}_f := \mathbf{k}_f$.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_f, \mathsf{ct})$**:** *Parse* $\mathsf{ct} = (\beta_0, \beta_1^\top, \beta_2^\top)$. *Compute*

$$\mathbf{H}_{f,\mathbf{x}} := \mathsf{EvalFX}(f, \mathbf{x}, \{\mathbf{A}_i\}_{i\in[\ell]}).$$

Let

$$\mu' = \beta_0 - \mathbf{k}_f^\top \begin{pmatrix} \beta_1 \\ \mathbf{H}_{f,\mathbf{x}}^\top \beta_1 \end{pmatrix}.$$

Output 1 if $|\mu'| \geq q/4$ *and* 0 *otherwise.*

To ensure security and (t-)correctness, the parameters need to satisfy the following conditions.

- $t\sigma(1 + 4m^2\rho(1 + \ell^{3/2}m(n\log q)^{O(d)})) < q/4$ for correctness.
- σ has to be chosen such that the hardness of the LWE assumption is ensured.
- $m \geq 6n\log q$ needs to hold for the $\mathsf{TrapGen}$ algorithm.
- $\rho \geq O(\sqrt{n\log q})\omega(\sqrt{\log m})$ is required for the success of the $\mathsf{SampleLeft}$ algorithm.
- $q > \sigma\Theta(\sqrt{\lambda})$ needs to hold to ensure the closeness of the truncated discrete Gaussian to its non-truncated counterpart.

For $N(\lambda) := \max\{\lambda, d(\lambda), \ell(\lambda), t(\lambda)\}$ and $\alpha \in (0,1)$, the following set of parameters satisfies the constraints above: $n = N^{2/\alpha}$, $m = n^2$, $q = m^{4N}$, $\sigma = 3\sqrt{n}$, $\rho = 2m$.

Proof (of Lemma 13). Let $\mathsf{ABE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be the ABE scheme as described in Construction 8. We define the following two algorithms.

$\mathsf{GenV}(\mathsf{mpk}, \mathbf{x})$**:** Let $\mathbf{H} := [[\mathbf{A}_1||\ldots||\mathbf{A}_\ell] + \mathbf{x}^\top \otimes \mathbf{G}] \in \mathbb{Z}_q^{n\times \ell m}$ and define

$$\begin{aligned}
\mathbf{v}_1 &:= \left(\left\lceil \frac{q}{2} \right\rceil, \mathbf{w}[1], \ldots, \mathbf{w}[n], \epsilon_{(\ell+1)m+1}\right)^\top, \\
\mathbf{v}_{i+1} &:= (0, \mathbf{A}[1][i], \ldots, \mathbf{A}[n][i], \epsilon_{i+1})^\top,\ i \in [m] \\
\mathbf{v}_{i+m+1} &:= (0, \mathbf{H}[1][i], \ldots, \mathbf{H}[n][i], \epsilon_{i+m+1})^\top,\ i \in [\ell m]
\end{aligned}$$

where ϵ_i is a vector of unity of size $(\ell+1)m+1$. Return $(\{\mathbf{v}_i\}_{i\in[(\ell+1)m+1]}, \mathbf{x})$.
$\mathsf{GenU}(\mathsf{mpk})$**:** Sample $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\mathbf{e}_0 \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma,\leq q}$, $\mathbf{e}_1 \leftarrow \mathcal{D}_{\mathbb{Z}^{m\ell},\sigma,\leq q}$ and $e' \leftarrow \mathcal{D}_{\mathbb{Z},\sigma,\leq q}$ Return

$$\mathbf{u} = (\mathbf{s}[1], \ldots, \mathbf{s}[n], e', \mathbf{e}_0[1], \ldots, \mathbf{e}_0[m], \mathbf{e}_1[1], \ldots, \mathbf{e}_1[\ell m])^\top \in \mathbb{Z}_q^{(\ell+1)m+n+1}.$$

To show that ABE has linear constructible ciphertexts, let us start by proving the security. Let

$\mathsf{Enc}'(\mathsf{mpk}, \mathbf{x}, \mu)$ Run

$$\begin{aligned}
(\{\mathbf{v}_i\}_{i\in[(\ell+1)m+1]}, \mathbf{x}) &\leftarrow \mathsf{GenV}(\mathsf{mpk}, \mathbf{x}), \\
\mathbf{u} &\leftarrow \mathsf{GenU}(\mathsf{mpk})
\end{aligned}$$

and set $\mathsf{ct}_i := (\mu||\mathbf{u}^\top)\mathbf{v}_i$ for all $i \in [(\ell+1)m+1]$. Return $(\{\mathsf{ct}_i\}_{i\in[(\ell+1)m+1]}, \mathbf{x})$.

Note that the security breaks down to the advantage of an adversary in distinguishing between a ciphertext produced by Enc and Enc'. But, by construction, $\beta_0 \equiv \mathsf{ct}_1, \beta_1^\top \equiv (\mathsf{ct}_2, \ldots, \mathsf{ct}_{m+1})$ and $\beta_2^\top \equiv (\mathsf{ct}_{m+2}, \ldots, \mathsf{ct}_{(\ell+1)m+1})$.

For the t-correctness, let $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$, $\mathbf{u}_j \leftarrow \mathsf{GenU}(\mathsf{mpk})$ for $j \in [t]$, $(\{\mathbf{v}_i\}_{i\in[(\ell+1)m+1]}, \mathbf{x}) \leftarrow \mathsf{GenV}(\mathsf{mpk}, \mathbf{x})$ and $\mu' \leftarrow \mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_f, (\{(\mu \parallel \sum_{j\in[t]} \mathbf{u}_j^\top) \cdot \mathbf{v}_i\}_{i\in[\ell]}, \mathbf{x}))$ such that $f(\mathbf{x}) = 0$. Let $\beta_1^\top = (\mathsf{ct}_2, \ldots, \mathsf{ct}_{m+1})$ and $\beta_2^\top = (\mathsf{ct}_{m+2}, \ldots, \mathsf{ct}_{(\ell+1)m+1})$. Then

$$\mu' = \mu\left\lceil \frac{q}{2} \right\rceil + \sum_{i\in[t]} e_i' - \sum_{i\in[t]} \mathbf{k}_f^\top \begin{pmatrix} \mathbf{e}_{0,i} \\ \mathbf{H}_{f,x}^\top \mathbf{e}_i \end{pmatrix}.$$

Bounding the error yields

$$\left\| \sum_{i\in[t]} e'_i - \sum_{i\in[t]} \mathbf{k}_f^\top \begin{pmatrix} \mathbf{e}_{0,i} \\ \mathbf{H}_{f,x}^\top \mathbf{e}_i \end{pmatrix} \right\| \leq t\sigma + 2m^{\frac{3}{2}}\rho\sqrt{2}(t(\sigma\sqrt{m} + \ell m(n\log q)^{O(d)}\sigma\sqrt{\ell m}))$$
$$\leq t\sigma(1 + 4m^2\rho(1 + \ell^{3/2}m(n\log q)^{O(d)})).$$

By our choice of parameters, this is smaller than $q/4$, which yields t-correctness. □

B.2 CP-ABE scheme with LCC

Let us consider the CP-ABE scheme by Katsumata et al. [KNYY20], whose access policies are inner products. At the heart of their CP-ABE scheme is a constrained pseudorandom function supporting inner products over some domain that is IPE-conforming. Let us first recall these definitions.

Definition 16 (Constrained Pseudorandom Function (CPRF)). *A constrained pseudorandom function for $(\mathcal{D}, \mathcal{R}, \mathcal{K}, \mathcal{C})$, with $\mathcal{D}$ and $\mathcal{R}$ being the range and domain of the PRF, $\mathcal{K}$ the PFR keys and $\mathcal{C} : \mathcal{D} \to \{0,1\}$ a family of circuits with polynomial bounded size, consists fof five PPT algorithms* $\mathsf{CPRF} = (\mathsf{cSetup}, \mathsf{cGen}, \mathsf{cEval}, \mathsf{cConstrain}, \mathsf{cConstrainEval})$.

$\mathsf{cSetup}(1^\lambda) \to mpk$**:** *On input the security parameter 1^λ, the algorithm outputs a master public key* mpk.
$\mathsf{cGen}(\mathsf{mpk}) \to \mathsf{K}$**:** *On input the master public key* mpk*, the key generation algorithm outputs a master key* $\mathsf{K} \in \mathcal{K}$.
$\mathsf{cEval}(\mathsf{mpk}, \mathsf{K}, x) \to r$**:** *On input the master public key* mpk*, a master key* K *and $x \in \mathcal{D}$, the evaluation algorithm outputs $r \in \mathcal{R}$.*
$\mathsf{cConstrain}(\mathsf{K}, C) \to \mathsf{K}_C^{\mathsf{con}}$ *On input a master key* K *and a circuit $C \in \mathcal{C}$ specifying a constraint, the constrained key generation algorithm outputs a constrained key* $\mathsf{K}_C^{\mathsf{con}}$.
$\mathsf{cConstrainEval}(\mathsf{mpk}, \mathsf{K}_C^{\mathsf{con}}, x) \to r$**:** *On input the master public key* mpk*, a constrained key* $\mathsf{K}_C^{\mathsf{con}}$ *and $x \in \mathcal{D}$, the constrained evaluation algorithm outputs $r \in \mathcal{R}$.*

Definition 17 (Partial linear evaluation [KNYY20]). *The algorithm* cEval *(resp.* $\mathsf{cConstrainEval}$*) can be divided into a linear part* $\mathsf{cEvalLin}$ *(resp.* $\mathsf{cConstrainEvalLin}$*) and a non-linear part* $\mathsf{cEvalNonLin}$ *(resp.* $\mathsf{cConstrainEvalNonLin}$*) with the following syntax:*

- $\mathsf{cEvalLin}(\mathsf{K}, \mathbf{x}) \to \mathsf{K}_{\mathbf{x}}^{\mathsf{int}} \in \mathbb{Z}^\xi$,
- $\mathsf{cEvalNonLin}(\mathsf{mpk}, \mathsf{K}_{\mathbf{x}}^{\mathsf{int}}, x) \to \mathsf{PRF}(\mathsf{K}, x)$,
- $\mathsf{cConstrainEvalLin}(\mathsf{K}_{\mathbf{y}}^{\mathsf{con}}, \mathbf{x}) \to \mathsf{K}_{\mathbf{y},vecx}^{\mathsf{con\text{-}int}} \in \mathbb{Z}^\xi$,
- $\mathsf{cConstrainEvalNonLin}(\mathsf{mpk}, \mathsf{K}_{\mathbf{y},\mathbf{x}}^{\mathsf{con\text{-}int}}, \mathbf{x}) \to \mathsf{PRF}(\mathsf{K}, \mathbf{x})$,

where the superscript int *stands for "intermediate key" and* $\mathsf{K}_{\mathbf{y}}^{\mathsf{con}}$ *denotes the constrained key for the inner-product constraint for vector* $\mathbf{y}$*. Specifically, we have:*

$$\mathsf{cEvalNonLin}(\mathsf{mpk}, \mathsf{cEvalLin}(\mathsf{K}, \mathbf{x}), \mathbf{x}) = \mathsf{cEval}(\mathsf{K}, \mathbf{x})$$

and

$$\mathsf{cConstrainEvalNonLin}(\mathsf{mpk}, \mathsf{cConstrainEvalLin}(\mathsf{K}, \mathbf{x}), \mathbf{x}) = \mathsf{cConstrainEval}(\mathsf{K}, \mathbf{x}).$$

We require the following:

1. $\mathsf{cEvalNonLin}$ *and* $\mathsf{cConstrainEvalNonLin}$ *are exactly the same algorithms, i.e., they are expressed identically as circuits.*
2. *For any $\mathbf{x}, \mathbf{y}$ such that $\langle \mathbf{x}, \mathbf{y} \rangle = 0$ and $\mathsf{K} \leftarrow \mathsf{cSetup}(\mathsf{mpk})$ where $\mathsf{mpk} \leftarrow \mathsf{cSetup}(1^\lambda)$, we have:*

$$\mathsf{cEvalLin}(K, x) = \mathsf{cConstrainEvalLin}(\mathsf{cConstrain}(\mathsf{K}, \mathbf{y}), \mathbf{x}),$$

or equivalently, $\mathsf{K}_{\mathbf{x}}^{\mathsf{int}} = \mathsf{K}_{\mathbf{y},\mathbf{x}}^{\mathsf{con\text{-}int}}$.

3. $\mathsf{K}, \mathsf{K}^{\mathsf{con}}_{\mathbf{y}}, \mathsf{K}^{\mathsf{int}}_{\mathbf{x}}$, *and* $\mathsf{K}^{\mathsf{con\text{-}int}}_{\mathbf{y},\mathbf{x}}$ *are integer vectors. Also, for any* $\mathbf{x}, \mathbf{y} \in \mathcal{D}$, *algorithms* $\mathsf{cConstrain}(\cdot, \mathbf{y})$, $\mathsf{cEvalLin}(\cdot, \mathbf{x})$, *and* $\mathsf{cConstrainEvalLin}(\cdot, \mathbf{x})$ *are linear functions over* $\mathbb{Z}$. *Moreover, their norms are at most* $\mathsf{poly}(\lambda, \ell, B)$.
4. *We have* $\|\mathsf{K}^{\mathsf{int}}_{\mathbf{x}}\|_\infty = \mathsf{poly}(\lambda, \ell, B)$ *where* $\mathsf{mpk} \leftarrow \mathsf{cSetup}(1^\lambda)$, $\mathsf{K} \leftarrow \mathsf{cSetup}(\mathsf{mpk})$, *and* $\mathsf{K}^{\mathsf{int}}_{\mathbf{x}} := \mathsf{cEvalLin}(\mathsf{K}, \mathbf{x})$.

For readability, as in [KNYY20, Tsa19], let $U^{\mathsf{lin}}_{\mathbf{k}\to\mathbf{x}}$, $U^{\mathsf{lin}}_{\mathbf{k}\to\mathbf{y}}$ and $U^{\mathsf{lin}}_{\mathbf{y}\to\mathbf{x}}$ be a linear functions computing $\mathsf{cEvalLin}(\cdot, \mathbf{x})$, $\mathsf{cConstrain}(\cdot, \mathbf{y})$ and $\mathsf{cConstrainEvalLin}(\cdot, \mathbf{x})$, respectively, and $U^{\mathsf{non\text{-}lin}}_{\mathbf{x}}$ be a function that computes $\mathsf{cEvalNonLin}(\mathsf{mpk}, \cdot, \mathbf{x})$.

Definition 18 (Key Simulation [KNYY20]). *Given a* PPT *adversary* $\mathcal{A}$ *against the key simulation security of* CPRF, *we define the experiment* $\mathsf{Exp}^{\mathsf{ks}\text{-}b}_{\mathsf{CPRF},\mathcal{A}}$ *as shown in Figure 10. Let* $\mathcal{CT}$ *be the collection of ciphertext queries. The adversary* $\mathcal{A}$ *is admissible with respect to* $\mathcal{CT}$, *denoted by* $\mathsf{adm}(\mathcal{A}) = 0$, *if for all* $\mathbf{x} \in \mathcal{CT}, \langle \mathbf{x}, \mathbf{y}^* \rangle \neq 0$. *Otherwise, we say that* $\mathcal{A}$ *is not admissible and write* $\mathsf{adm}(\mathcal{A}) = 1$.

CPRF *is said to be key simulation secure if* $\mathsf{Exp}^{\mathsf{ks}\text{-}0}_{\mathsf{CPRF},\mathcal{A}} \approx_c \mathsf{Exp}^{\mathsf{ks}\text{-}1}_{\mathsf{CPRF},\mathcal{A}}$.

$\mathsf{Exp}^{\mathsf{ks}\text{-}b}_{\mathsf{CPRF},\mathcal{A}}$

$\mathcal{CT} := \emptyset$
$\mathsf{mpk} \leftarrow \mathsf{cSetup}(1^\lambda)$
$\mathsf{K} \leftarrow \mathsf{cGen}(\mathsf{mpk})$
$\mathbf{y}^* \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{cEval}}(\mathsf{mpk})$
$\widetilde{\mathsf{K}} \leftarrow \mathsf{cGen}(\mathsf{mpk})$
$\mathsf{K}_0 := \widetilde{\mathsf{K}}$
$\mathsf{K}_1 := \mathsf{K}$
$\widetilde{\mathsf{K}}^{\mathsf{con}}_{\mathbf{y}^*} \leftarrow \mathsf{cConstrain}(\mathsf{K}_b, \mathbf{y}^*)$
$b' \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{cEval}}(\widetilde{\mathsf{K}}^{\mathsf{con}}_{\mathbf{y}^*})$
if $\mathsf{adm}(\mathcal{A}) = 0$ **then**
  return $\beta := (b' \stackrel{?}{=} b)$
return $\beta \xleftarrow{\$} \{0, 1\}$

$\mathcal{O}\mathsf{cEval}(\mathbf{x})$

$\mathcal{CT} := \mathcal{CT} \cup \{\mathbf{x}\}$
$\mathbf{r} \leftarrow \mathsf{cEval}(\mathsf{mpk}, \mathsf{K}, \mathbf{x})$
return $\mathbf{r}$

Fig. 10. Security game $\mathsf{Exp}^{\mathsf{ks}\text{-}b}_{\mathsf{CPRF},\mathcal{A}}$ for Definition 18.

Definition 19 (Uniformity [KNYY20]). *For all* $\mathbf{x} \in \mathcal{D}$ *and* $\mathbf{r} \in \mathcal{R}$, *we have*

$$\Pr(\mathsf{cEval}(\mathsf{K}, \mathbf{x}) = \mathbf{r} |\ \mathsf{mpk} \leftarrow \mathsf{cSetup}(1^\lambda), \mathsf{K} \leftarrow \mathsf{cGen}(\mathsf{mpk})) = 1/|\mathcal{R}|.$$

Definition 20 (IP-Conforming). *A CPRF scheme* $\mathsf{CPRF} = (\mathsf{cSetup}, \mathsf{cGen}, \mathsf{cEval}, \mathsf{cConstrain}, \mathsf{cConstrainEval})$ *that supports inner products over some domain* $\mathcal{D} := [-B, B]^\ell \subset \mathbb{Z}^\ell$ *is said to be IPE-conforming if it satisfies partial linear evaluation, key simulation security and uniformity.*

For readability, we use the following notation as in [KNYY20, Tsa19] For a (homogeneous) linear function $f : \mathbb{Z}^\ell \to \mathbb{Z}^k$, let $\mathbf{M}_f \in \mathbb{Z}^{\ell \times k}$ be the unique matrix that represents f, i.e., $f(\mathbf{x})^\top = \mathbf{x}^\top \mathbf{M}_f$.

Lemma 22. *There exist efficient deterministic algorithms* EvalLin *such that for all* $n, m, q, \ell \in \mathbb{N}$, *for any linear function* $f : \mathbb{Z}^\ell \to \mathbb{Z}^k$, *input* $\mathbf{x} \in \mathbb{Z}^\ell$ *and matrix* $\mathbf{A} \in \mathbb{Z}^{n \times m\ell}$, *the output* $\overline{\mathbf{M}}_f := \mathsf{EvalLin}(f)$ *is in* $\mathbb{Z}^{m\ell \times mk}$, *satisfies* $\left\|\overline{\mathbf{M}}_f\right\|_\infty = \max\{1, \|f\|_\infty\}$ *and*

$$[\mathbf{A} - \mathbf{x}^\top \otimes \mathbf{G}]\overline{\mathbf{M}}_f = \mathbf{A}\overline{\mathbf{M}}_f - f(\mathbf{x})^\top \otimes \mathbf{G} \mod q.$$

Lemma 23. *There exist efficient deterministic algorithms* $\mathsf{EvalF}^{\mathsf{bd}}$, $\mathsf{EvalFX}^{\mathsf{bd}}$ *such that for all* $n, m, \ell, M \in \mathbb{N}$ *with* $m \geq n\lceil \log q \rceil$, *prime* $q > 2M+1$, *for any function* $f : [-M, M]^\ell \to \{0,1\}^k$ *that can be expressed as an efficient boolean circuit* $\widetilde{f} : \{0,1\}^{\ell\lceil \log q \rceil} \to \{0,1\}^k$ *of depth* d, *for every* $\mathbf{x} \in [-M, M]^\ell$ *and for any matrix* $\mathbf{A} \in \mathbb{Z}^{n \times m\ell}$, *the outputs* $\mathbf{H} := \mathsf{EvalF}^{\mathsf{bd}}(1^M, f, \mathbf{A})$ *and* $\widehat{\mathbf{H}} := \mathsf{EvalFX}^{\mathsf{bd}}(1^M, f, x, \mathbf{A})$ *are both in* $\mathbb{Z}^{m\ell \times mk}$, *satisfy*

$$\|\mathbf{H}\|_\infty, ||\widehat{\mathbf{H}}||_\infty \leq \ell\lceil \log q \rceil (2mM)^{d+2M+2}$$

and

$$[\mathbf{A} - \mathbf{x}^\top \otimes \mathbf{G}]\widehat{\mathbf{H}} = \mathbf{A}\mathbf{H} - f(\mathbf{x})^\top \otimes \mathbf{G} \mod q.$$

Let $\mathsf{Eq}_{\mathbf{r}} : \{0,1\}^\eta \to \{0,1\}$ be the circuit that on input $\widetilde{\mathbf{r}}$ returns 1 if and only if $\mathbf{r} = \widetilde{\mathbf{r}}$. As $\mathsf{K}, \mathsf{K}^{\mathsf{int}}_{\mathbf{x}}$ and $\mathsf{K}^{\mathsf{con}}_{\mathbf{y}}$ are vectors in the construction, we will denote them as such, i.e., $\mathbf{k}, \mathbf{k}^{\mathsf{int}}_{\mathbf{x}}$ and $\mathbf{k}^{\mathsf{con}}_{\mathbf{y}}$. Moreover, as the policies f are inner products, we identify them with the vector $\mathbf{y}$. We will now recall the CP-ABE scheme from [KNYY20] with truncated errors.

Construction 9. *Let* $\mathsf{CPRF} = (\mathsf{cSetup}, \mathsf{cGen}, \mathsf{cEval}, \mathsf{cConstrain}, \mathsf{cConstrainEval})$ *be a CPRF scheme with domain* $\mathcal{D} = [-B, B]^\ell$ *and range* $\mathcal{R} \subset \{0,1\}^\eta$, $\mathbf{k}, \mathbf{k}^{\mathsf{int}}_{\mathbf{x}}$ *and* $\mathbf{k}^{\mathsf{con}}_{\mathbf{y}}$ *integer vectors of dimension* $s = s(\lambda)$ *that fulfills the IPE confirming property. Let* $d = d(\lambda)$ *be an upper bound on the depth of the circuits* $U^{\mathsf{non\text{-}lin}}_{\mathbf{x}}$ *and* $M(\lambda)$ *be an upper bound on* $\left\|\mathsf{K}^{\mathsf{int}}_{\mathbf{x}}\right\|_\infty$ *and the norms of* $U^{\mathsf{lin}}_{\mathbf{k}\to\mathbf{x}}, U^{\mathsf{lin}}_{\mathbf{k}\to\mathbf{y}}$ *and* $U^{\mathsf{non\text{-}lin}}_{\mathbf{x}}$.

$\mathsf{Setup}(1^\lambda)$**:** *Generate* $\mathsf{cmpk} \leftarrow \mathsf{cSetup}(1^\lambda), \leftarrow\mathsf{cGen}(\mathsf{cmpk})$, *sample* $(\mathbf{B}, \mathbf{T}_{\mathbf{B}}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$, $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times ms}$ *and* $\mathbf{w} \xleftarrow{\$} \mathbb{Z}_q^n$. *Output* $\mathsf{mpk} = (\mathbf{B}, \mathbf{A}, \mathbf{w}, \mathsf{cmpk})$ *and* $\mathsf{msk} = (\mathbf{T}_{\mathbf{B}}, \mathbf{k}, \mathsf{mpk})$.

$\mathsf{Enc}(\mathsf{mpk}, \mathbf{y}, \mu \in \{0,1\})$**:** *Run* $\widetilde{\mathbf{k}} \leftarrow \mathsf{cGen}(\mathsf{cmpk})$ *and sample* $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $e_0 \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\sigma,\leq q}$, $\mathbf{e}_1 \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^m,\sigma,\leq q}$ *and* $\mathbf{e}_2 \xleftarrow{\$} [-\Gamma, \Gamma]^{ms}$. *Compute* $\widetilde{\mathbf{k}}^{\mathsf{con}}_{\mathbf{y}} \leftarrow \mathsf{cConstrain}(\widetilde{\mathbf{k}}, C_{\mathbf{y}})$, $\overline{\mathbf{M}}_{\mathbf{k}\to\mathbf{y}} \leftarrow \mathsf{EvalLin}(U^{\mathsf{lin}}_{\mathbf{k}\to\mathbf{y}})$ *and let* $\mathbf{A}^{\mathsf{con}}_{\mathbf{y}} = \mathbf{A}\overline{\mathbf{M}}_{\mathbf{k}\to\mathbf{y}}$. *Set*

$$\beta_0 = \mathbf{w}^\top \mathbf{s} + e_0, \ \beta_1^\top = \mathbf{s}^\top \mathbf{B} + \mathbf{e}_1^\top, \ \beta_2^\top = \mathbf{s}^\top [\mathbf{A}^{\mathsf{con}}_{\mathbf{y}} - (\widetilde{\mathbf{k}}^{\mathsf{con}}_{\mathbf{y}})^\top) \otimes \mathbf{G}] + \mathbf{e}_2^\top.$$

Return $\mathsf{ct} = (\beta_0, \beta_1^\top, \beta_2^\top, \mathbf{y}, \widetilde{\mathbf{k}}^{\mathsf{con}}_{\mathbf{y}})$.

$\mathsf{KeyGen}(\mathsf{msk}, \mathbf{x})$**:** *Compute* $\mathbf{r} = \mathsf{cEval}(\mathbf{k}, \mathbf{x})$,

$$\begin{aligned} \overline{\mathbf{M}}_{\mathbf{k}\to\mathbf{y}} &\leftarrow \mathsf{EvalLin}(U^{\mathsf{lin}}_{\mathbf{k}\to\mathbf{x}}), & \mathbf{A}^{\mathsf{int}}_{\mathbf{x}} &:= \mathbf{A}\overline{\mathbf{M}}_{\mathbf{k}\to\mathbf{y}}, \\ \mathbf{H}_{\mathbf{x}} &\leftarrow \mathsf{EvalF}^{\mathsf{bd}}(1^M, U^{\mathsf{non\text{-}lin}}_{\mathbf{x}}, \mathbf{A}^{\mathsf{int}}_{\mathbf{x}}), & \mathbf{A}^{\mathsf{eval}}_{\mathbf{x}} &:= \mathbf{A}^{\mathsf{int}}_{\mathbf{x}}\mathbf{H}_{\mathbf{x}}, \\ \mathbf{H}_{\mathbf{r}} &\leftarrow \mathsf{EvalF}(\mathsf{Eq}_{\mathbf{r}}, \mathbf{A}^{\mathsf{eval}}_{\mathbf{x}}), & \mathbf{A}^{\mathsf{eq}}_{\mathbf{x},\mathbf{r}} &:= \mathbf{A}^{\mathsf{eval}}_{\mathbf{x}}\mathbf{H}_{\mathbf{r}}. \end{aligned}$$

Parse $\mathbf{A}^{\mathsf{eq}}_{\mathbf{x},\mathbf{r}} = [\mathbf{A}^{\mathsf{eq}}_{\mathbf{x},\mathbf{r},0} || \mathbf{A}^{\mathsf{eq}}_{\mathbf{x},\mathbf{r},1}]$ *with* $\mathbf{A}^{\mathsf{eq}}_{\mathbf{x},\mathbf{r},i} \in \mathbb{Z}_q^{n \times m}, i \in \{0,1\}$, *run* $\mathbf{z} \leftarrow \mathsf{SampleLeft}(\mathbf{B}, \mathbf{A}^{\mathsf{eq}}_{\mathbf{x},\mathbf{r},1}, \mathbf{T}_{\mathbf{B}}, \mathbf{w}, \rho)$ *and output* $\mathsf{sk}_{\mathbf{x}} = (\mathbf{x}, \mathbf{r}, \mathbf{z})$.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_{\mathbf{x}}, \mathsf{ct})$**:** *Parse* $\mathsf{sk}_{\mathbf{x}} = (\mathbf{x}, \mathbf{r}, \mathbf{z})$ *and* $\mathsf{ct} = (\mathbf{y}^\top, (\widetilde{\mathbf{k}}^{\mathsf{con}}_{\mathbf{y}})^\top, \beta_0, \beta_1^\top, \beta_2^\top)$. *Compute* $\widetilde{\mathbf{k}}^{\mathsf{int}}_{\mathbf{x}} := U^{\mathsf{lin}}_{\mathbf{y}\to\mathbf{x}}(\widetilde{\mathbf{k}}^{\mathsf{con}}_{\mathbf{y}})$ *and* $\widetilde{\mathbf{r}} := U^{\mathsf{non\text{-}lin}}_{\mathbf{x}}(\widetilde{\mathbf{k}}^{\mathsf{int}}_{\mathbf{x}})$. *Abort if* $\mathbf{r} = \widetilde{\mathbf{r}}$. *Otherwise, compute*

$$\begin{aligned} &\overline{\mathbf{M}}_{\mathbf{y}\to\mathbf{x}} \leftarrow \mathsf{EvalLin}(U^{\mathsf{lin}}_{\mathbf{y}\to\mathbf{x}}), \widehat{\mathbf{H}}_{\mathbf{x}} \leftarrow \mathsf{EvalFX}^{\mathsf{bd}}(1^M, U^{\mathsf{non\text{-}lin}}_{\mathbf{x}}, \widetilde{\mathbf{k}}^{\mathsf{int}}_{\mathbf{x}}, \mathbf{A}^{\mathsf{int}}_{\mathbf{x}}), \\ &\widehat{\mathbf{H}}_{\mathbf{r}} \leftarrow \mathsf{EvalFX}(\mathsf{Eq}_{\mathbf{r}}, \widetilde{\mathbf{r}}, \mathbf{A}^{\mathsf{eval}}_{\mathbf{x}}), \end{aligned}$$

where $\mathbf{A}^{\mathsf{int}}_{\mathbf{x}}$ *and* $\mathbf{A}^{\mathsf{eval}}_{\mathbf{x}}$ *are computed as in* KeyGen. *Let*

$$\mu' := \beta_0 - [\beta_1^\top || \beta_2^\top \overline{\mathbf{M}}_{\mathbf{y}\to\mathbf{x}} \widehat{\mathbf{H}}_{\mathbf{x}} \widehat{\mathbf{H}}_{\mathbf{r}} [\mathbf{0}_m || \mathbf{I}_m]^\top]\mathbf{z}.$$

Output 1 if $|\mu'| \geq q/4$ *and* 0 *otherwise.*

Fix $\ell = \ell(\lambda), B = B(\lambda)$, and define based on that $s = s(\lambda), M = M(\lambda), \eta = \eta(\lambda)$ and $d = d(\lambda)$, which are all polynomial in λ. To ensure security and correctness, we choose the parameters as in [KNYY20], with the additional requirements that

– $q > 12t\rho\Gamma m^4 s^3 \eta^2 (\lceil \log q \rceil)^2 (2mM)^{2d+4M+5}$ for correctness,
– $q > \sigma\Theta(\sqrt{\lambda})$ due to the use of truncated errors.

These can be easily incorporated without infringing any of the other constraints under the assumption that t is at most polynomial in λ. Hence, if we define $N(\lambda) := \max\{\lambda, s(\lambda), M(\lambda), \eta(\lambda), d(\lambda), t(\lambda) \in\}$ and $\alpha \in (0,1)$, then the following set of parameters can be used: $n = N^{2/\alpha}$, $m = n^2$, $q = (2mN)^{15N}$, $\sigma = 3\sqrt{n}$, $\rho = (2mN)^{7N}$, $\Gamma = (2mN)^N$.

Proof (of Lemma 14). Let $\mathsf{ABE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be the ABE scheme as described in Construction 9. We define the following two algorithms.

$\mathsf{GenU}(\mathsf{mpk})$**:** Sample $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $e_0 \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\sigma,\leq q}$, $\mathbf{e}_1 \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^m,\sigma,\leq q}$ and $\mathbf{e}_2 \xleftarrow{\$} [-\Gamma,\Gamma]^{ms}$. Return

$$\mathbf{u} = (\mathbf{s}[1], \ldots, \mathbf{s}[n], e_0, \mathbf{e}_1[1], \ldots, \mathbf{e}_1[m], \mathbf{e}_2[1], \ldots, \mathbf{e}_2[ms])^\top \in \mathbb{Z}_p^{n+m(s+1)+1}.$$

$\mathsf{GenV}(\mathsf{mpk}, \mathbf{y})$**:** Compute $\widetilde{\mathbf{k}}_{\mathbf{y}}^{\mathsf{con}} \leftarrow \mathsf{cConstrain}(\widetilde{\mathbf{k}}, C_{\mathbf{y}})$, $\overline{\mathbf{M}}_{\mathbf{k}\to\mathbf{y}} \leftarrow \mathsf{EvalLin}(U^{\mathsf{lin}}_{\mathbf{k}\to\mathbf{y}})$ and let $\mathbf{A}_{\mathbf{y}}^{\mathsf{con}} = \mathbf{A}\overline{\mathbf{M}}_{\mathbf{k}\to\mathbf{y}}$, $\mathbf{H} = [\mathbf{A}_{\mathbf{y}}^{\mathsf{con}} - (\widetilde{\mathbf{k}}_{\mathbf{y}}^{\mathsf{con}})^\top \otimes \mathbf{G}]$. Set

$$\begin{aligned}
\mathbf{v}_1 &= (\lfloor q/2 \rfloor, \mathbf{w}[1], \ldots, \mathbf{w}[n], \boldsymbol{\epsilon}_1)\,, \\
\mathbf{v}_{i+1} &= (0, \mathbf{B}[1][i], \ldots, \mathbf{B}[n][i], \boldsymbol{\epsilon}_{i+1}) \qquad i \in [m], \\
\mathbf{v}_{i+m+1} &= (0, \mathbf{H}[1][i], \ldots, \mathbf{H}[n][i], \boldsymbol{\epsilon}_{i+m+1}) \quad i \in [ms],
\end{aligned}$$

where $\boldsymbol{\epsilon}_i$ denotes the i^{th} canonical vector of size $m(s+1)+1$. Return

$$(\{\mathbf{v}_i\}_{i\in[m(s+1)+1]}, \mathbf{x}, \widetilde{\mathbf{k}}_{\mathbf{y}}^{\mathsf{con}}).$$

Consider the encryption algorithm.

$\mathsf{Enc}'(\mathsf{mpk}, \mathbf{y}, \mu)$**:** Run

$$\begin{aligned}
(\{\mathbf{v}_i\}_{i\in[m(s+1)+1]}, \mathbf{x}, \widetilde{\mathbf{k}}_{\mathbf{y}}^{\mathsf{con}}) &\leftarrow \mathsf{GenV}(\mathsf{mpk}, \mathbf{x}), \\
\mathbf{u} &\leftarrow \mathsf{GenU}(\mathsf{mpk}),
\end{aligned}$$

and set $\mathsf{ct}_i := (\mu || \mathbf{u}^\top)\mathbf{v}_i$ for all $i \in [m(s+1)+1]$.
Return $(\{\mathsf{ct}_i\}_{i\in[m(s+1)+1]}, \mathbf{y}, \widetilde{\mathbf{k}}_{\mathbf{y}}^{\mathsf{con}})$.

Note that the security breaks down to the advantage of an adversary in distinguishing between a ciphertext produced by Enc and Enc'. But, by construction, $\beta_0 \equiv \mathsf{ct}_1, \beta_1^\top \equiv (\mathsf{ct}_2, \ldots, \mathsf{ct}_{m+1})$ and $\beta_2^\top \equiv (\mathsf{ct}_{m+1}, \ldots, \mathsf{ct}_{m(s+1)+1})$.

For the t-correctness, let $\mathbf{u}' = \sum_{i\in[t]} \mathbf{u}_i$ with $\mathbf{u}_i \leftarrow \mathsf{GenU}(\mathsf{mpk})$ for $i \in [t]$, $\mathsf{sk}_{\mathbf{x}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathbf{x})$, $(\{\mathbf{v}_i\}_{i\in[m(s+1)+1]}, \mathbf{x}, \widetilde{\mathbf{k}}_{\mathbf{y}}^{\mathsf{con}}) \leftarrow \mathsf{GenV}(\mathsf{mpk}, \mathbf{x})$ and

$$\mu' \leftarrow \mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_{\mathbf{x}}, (\{(\mu || \mathbf{u}'^\top)\mathbf{v}_i\}_{i\in[m(s+1)+1]}, \mathbf{y}, \widetilde{\mathbf{k}}_{\mathbf{y}}^{\mathsf{con}})).$$

Note that the usage of $\mathbf{u}'$ instead of a regularly sampled $\mathbf{u}$ only changes the error terms and not the evaluation algorithm. Thus, by the same arguments as in [KNYY20], the magnitude of the error after decryption is given by

$$\begin{aligned}
&\left\| e_0' - [(e_1')^\top || (e_2')^\top \overline{\mathbf{M}}_{\mathbf{y}\to\mathbf{x}} \widehat{\mathbf{H}}_{\mathbf{x}} \widehat{\mathbf{H}}_{\mathbf{r}} [\mathbf{0}_m || \mathbf{I}_m]^\top] \mathbf{z} \right\|_\infty \\
&\leq \|e_0'\| + 2m^4 s^2 \eta \|\mathbf{e}_1'\|_\infty \left\|\overline{\mathbf{M}}_{\mathbf{y}\to\mathbf{x}}\right\|_\infty \left\|\widehat{\mathbf{H}}_{\mathbf{x}}\right\|_\infty \left\|\widehat{\mathbf{H}}_{\mathbf{r}}\right\|_\infty \|\mathbf{z}\|_\infty \\
&\leq 3t\rho\Gamma m^4 s^3 \eta^2 (\lceil \log q \rceil)^2 (2mM)^{2d+4M+5} \\
&< q/4,
\end{aligned}$$

by our choice of parameters, where we used that we know the distribution of the summand in $\mathbf{u}'$ and Lemma 23. □